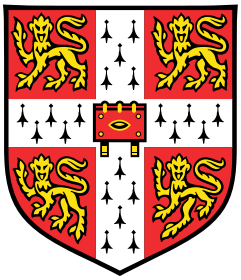
The Roles of Language Models and Hierarchical Models in Neural Sequence-to-Sequence Prediction



Felix Stahlberg

Supervisor: Prof. Bill Byrne

Advisor: Prof. Phil Woodland

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Doctor of Philosophy

Queens' College September 2019

Dedicated to Šepići.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Felix Stahlberg
September 2019

Acknowledgements

According to the academic code of conduct it is imperative to acknowledge my supervisor, Prof. Bill Byrne. Since acknowledging one's supervisor with flowery language is such a common thing to do, I find it difficult to sufficiently express my sincere and deep gratitude towards Bill in this context. There is no question that our many and long discussions were not only very enjoyable but also made me a much better researcher. Bill mastered the fine balance between guidance and freedom for me perfectly, giving me enough space and trust for pursuing my own (sometimes abstruse) research ideas while staying engaged to identify connections to other lines of research and alternative ways to go forward. Bill also taught me the value of formal rigorousness, and ensured that I have access to all the resources I needed for conducting and communicating my research – whether financial or computational.

I am also thankful to my co-authors Danielle Saunders from the Cambridge University, and Gonzalo Iglesias, Eva Hasler, and Adrià de Gispert from SDL Research. The scope of my PhD studies would have been much more limited without these fruitful collaborations. This thesis also draws from my two research internships – one at Google NY and one in the AML machine translation group at Facebook. I thank both teams for making me feel welcome, especially my hosts Brian Roark (Google), Richard Sproat (Google), and James Cross (Facebook).

On a personal note, I would like to thank my wife, Barbara Stahlberg, for her love, and constant support and encouragement. I also thank the Šepić family for (now officially) welcoming me in their midst last year, and Aarón Sanchez for always looking over my shoulder.

I was financially supported by the U.K. Engineering and Physical Sciences Research Council (EPSRC grant EP/L027623/1). Some of my work has been performed using resources provided by the Cambridge Tier-2 system operated by the University of Cambridge Research Computing Service[1] funded by EPSRC Tier-2 capital grant EP/P020259/1. I hope that, despite shifting political tides in the U.K., the research council will be able to continue to open doors for European students like it did for me.

[1] http://www.hpc.cam.ac.uk

Abstract

With the advent of deep learning, research in many areas of machine learning is converging towards the same set of methods and models. For example, long short-term memory networks are not only popular for various tasks in natural language processing (NLP) such as speech recognition, machine translation, handwriting recognition, syntactic parsing, etc., but they are also applicable to seemingly unrelated fields such as robot control, time series prediction, and bioinformatics. Recent advances in contextual word embeddings like BERT boast with achieving state-of-the-art results on 11 NLP tasks with the same model. Before deep learning, a speech recognizer and a syntactic parser used to have little in common as systems were much more tailored towards the task at hand.

At the core of this development is the tendency to view each task as yet another data mapping problem, neglecting the particular characteristics and (soft) requirements tasks often have in practice. This often goes along with a sharp break of deep learning methods with previous research in the specific area. This work can be understood as an antithesis to this paradigm. We show how traditional symbolic statistical machine translation models can still improve neural machine translation (NMT) while reducing the risk for common pathologies of NMT such as hallucinations and neologisms. Other external symbolic models such as spell checkers and morphology databases help neural grammatical error correction. We also focus on language models that often do not play a role in vanilla end-to-end approaches and apply them in different ways to word reordering, grammatical error correction, low-resource NMT, and document-level NMT. Finally, we demonstrate the benefit of hierarchical models in sequence-to-sequence prediction. Hand-engineered covering grammars are effective in preventing catastrophic errors in neural text normalization systems. Our operation sequence model for interpretable NMT represents translation as a series of actions that modify the translation state, and can also be seen as derivation in a formal grammar.

Table of contents

List of figures xvii

List of tables xxiii

Nomenclature xxix

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Contributions . . . 3
- 1.3 Organization of this Thesis . . . 5
- 1.4 Notations . . . 5

2 Statistical Machine Translation and Symbolic Models 7
- 2.1 Motivation . . . 7
- 2.2 N-gram Language Models . . . 8
- 2.3 Word-based Models . . . 8
- 2.4 Word Alignments . . . 9
- 2.5 Phrase-based Translation . . . 10
- 2.6 Weighted Finite State Transducers . . . 11
  - 2.6.1 Formalisms . . . 12
  - 2.6.2 Operations on WFSTs . . . 14
- 2.7 Hierarchical Phrase-based Machine Translation . . . 15
  - 2.7.1 Synchronous Grammars . . . 15
  - 2.7.2 FST-based Hierarchical Translation . . . 16
- 2.8 Conclusion . . . 17

3 Neural Machine Translation 19
- 3.1 Motivation . . . 19

3.2 Word Embeddings . . . 20
3.3 Phrase Embeddings . . . 21
3.4 Sentence Embeddings . . . 22
3.5 Encoder-Decoder Networks with Fixed Length Encodings . . . 23
3.6 Attentional Encoder-Decoder Networks . . . 26
3.6.1 Attention . . . 26
3.6.2 Attention Masks and Padding . . . 29
3.6.3 Recurrent Neural Machine Translation . . . 30
3.6.4 Convolutional Neural Machine Translation . . . 32
3.6.5 Self-attention-based Neural Machine Translation . . . 35
3.6.6 Comparison of the Fundamental Architectures . . . 37
3.7 Neural Machine Translation Decoding . . . 39
3.7.1 The Search Problem in NMT . . . 39
3.7.2 Greedy and Beam Search . . . 39
3.7.3 Formal Description of Decoding for the RNNsearch Model . . . 40
3.7.4 Ensembling . . . 42
3.7.5 Decoding Direction . . . 44
3.7.6 Efficiency . . . 45
3.7.7 Generating Diverse Translations . . . 46
3.7.8 Simultaneous Translation . . . 47
3.8 Open Vocabulary Neural Machine Translation . . . 48
3.8.1 Using Large Output Vocabularies . . . 48
3.8.2 Character-based NMT . . . 50
3.8.3 Subword-unit-based NMT . . . 52
3.8.4 Words, Subwords, or Characters? . . . 54
3.9 Using Monolingual Training Data . . . 55
3.10 NMT Model Errors . . . 57
3.10.1 Sentence Length . . . 57
3.11 NMT Training . . . 61
3.11.1 Cross-entropy Training . . . 62
3.11.2 Training Deep Architectures . . . 64
3.11.3 Regularization . . . 64
3.11.4 Large Batch Training . . . 66
3.11.5 Reinforcement Learning . . . 67
3.11.6 Dual Supervised Learning . . . 68
3.11.7 Adversarial Training . . . 68

3.12 Explainable Neural Machine Translation . . . 69
3.12.1 Post-hoc Interpretability . . . 69
3.12.2 Model-intrinsic Interpretability . . . 69
3.12.3 Confidence Estimation in Translation . . . 70
3.12.4 Word Alignment in Neural Machine Translation . . . 70
3.13 Alternative NMT Architectures . . . 71
3.13.1 Extensions to the Transformer Architecture . . . 71
3.13.2 Advanced Attention Models . . . 72
3.13.3 Memory-augmented Neural Networks . . . 73
3.13.4 Beyond Encoder-decoder Networks . . . 74
3.14 Data Sparsity . . . 75
3.14.1 Corpus Filtering . . . 75
3.14.2 Domain Adaptation . . . 76
3.14.3 Low-resource NMT . . . 77
3.14.4 Unsupervised NMT . . . 77
3.15 Multilingual NMT . . . 78
3.16 NMT Model Size . . . 79
3.17 NMT with Extended Context . . . 80
3.17.1 Multimodal NMT . . . 80
3.17.2 Tree-based NMT . . . 80
3.17.3 NMT with Graph Structures as Input . . . 81
3.17.4 Document-level Translation . . . 82
3.18 Conclusion . . . 82

4 SMT-NMT Hybrid Systems 85
4.1 Motivation . . . 85
4.2 Related Work . . . 86
4.3 Syntactically Guided Neural Machine Translation . . . 88
4.3.1 Hiero Predictive Posteriors . . . 90
4.3.2 FST-based Constrained NMT Decoding . . . 91
4.3.3 Experiments . . . 92
4.4 UCAM@WMT16: Combination Via Edit Distance Transducer . . . 97
4.4.1 The Edit Distance Transducer . . . 98
4.4.2 Loose Coupling of Hiero and NMT . . . 98
4.4.3 Experiments . . . 102
4.5 MBR-based Hybrid Machine Translation . . . 105
4.5.1 Combining NMT and SMT by Minimizing the Lattice Bayes-risk . . . 106

4.5.2 Left-to-right Decoding . . . 107
4.5.3 Efficient *n*-gram Posterior Calculation . . . 107
4.5.4 Relation to MBR Decoding . . . 108
4.5.5 Subword-level MBR . . . 108
4.5.6 Experiments . . . 109
4.6 UCAM@WMT18: MBR-based System Combination of Multiple Models . . . 112
4.6.1 Multiple System Combination . . . 112
4.6.2 System Setup . . . 114
4.6.3 Results . . . 118
4.7 Conclusion . . . 120

5 SGNMT – A Flexible NMT Decoding Platform 123
5.1 Motivation . . . 123
5.2 NMT Toolkits . . . 124
5.3 Predictors . . . 126
5.3.1 Example Predictor Constellations . . . 130
5.3.2 Predictor Wrappers . . . 131
5.4 Decoders . . . 132
5.4.1 Neural Decoding as Shortest Path Search . . . 134
5.4.2 NMT Batch Decoding . . . 138
5.4.3 Exact Inference in NMT . . . 139
5.4.4 Using Exact Inference to Characterize NMT Search Errors and Model Errors . . . 141
5.5 Output Formats . . . 146
5.6 Tuning Predictor Weights . . . 146
5.7 Applications . . . 148
5.7.1 SGNMT for Research . . . 148
5.7.2 SGNMT for Teaching . . . 149
5.7.3 SGNMT in the Industry . . . 150
5.8 Conclusion . . . 150

6 Unfolding and Shrinking Neural Machine Translation Ensembles 151
6.1 Motivation . . . 151
6.2 Unfolding Ensembles into a Single Large Neural Network . . . 153
6.3 Shrinking the Unfolded Network . . . 155
6.3.1 Data-Free Neuron Removal . . . 155
6.3.2 Data-Bound Neuron Removal . . . 158

6.3.3 Shrinking Embedding Layers with SVD . . . 159
6.4 Results . . . 159
6.5 Probabilistic Interpretation of Data-Free and Data-Bound Shrinking . . . 164
6.6 Conclusion . . . 166

7 The Role of Language Models in Neural Sequence-to-Sequence Prediction 169
7.1 Motivation . . . 169
7.2 Neural Word Reordering Using Language Models . . . 170
7.2.1 Bag-to-sequence Modeling with Attentional Neural Networks . . . 170
7.2.2 Search . . . 172
7.2.3 Results . . . 172
7.3 Neural Grammatical Error Correction Using FSTs . . . 174
7.3.1 Constructing the Hypothesis Space . . . 175
7.3.2 Experiments . . . 179
7.3.3 UCAM@BEA19: FST-based GEC in Shared Task Submissions . . . 182
7.4 Simple Fusion: Using Language Models for NMT Training . . . 185
7.4.1 Translation Model Training under Language Model Predictions . . . 186
7.4.2 Experimental Setup . . . 187
7.4.3 Results . . . 189
7.4.4 Discussion and Analysis . . . 189
7.5 UCAM@WMT19: Document-level Language Models for Translation . . . 195
7.5.1 Document-level Language Modelling . . . 195
7.5.2 Results . . . 197
7.6 Conclusion . . . 200

8 The Role of Hierarchical Models in Neural Sequence-to-Sequence Prediction 201
8.1 Motivation . . . 201
8.2 Syntax-based NMT with Multi-representation Ensembles . . . 202
8.3 Neural Text Normalization Under Covering Grammars . . . 204
8.3.1 Neural Text Normalization Models . . . 205
8.3.2 Unconstrained Neural Text Normalization . . . 209
8.3.3 Constraining Neural Text Normalization with Covering Grammars . . . 211
8.4 Operation Sequence Neural Machine Translation . . . 214
8.4.1 A Neural Operation Sequence Model . . . 215
8.4.2 OSNMT Represents Alignments . . . 216
8.4.3 OSNMT Represents Hierarchical Structure . . . 218
8.4.4 Comparison to the OSM for SMT . . . 221

8.4.5 Training . . . 222
8.4.6 Results . . . 222
8.4.7 Related Work . . . 229
8.4.8 Future Work . . . 230
8.5 Conclusion . . . 230

9 Conclusion **231**
9.1 Language Models . . . 231
9.2 Hierarchical Models . . . 233
9.3 Implementations . . . 233
9.4 Final Remarks . . . 234

References **235**

Appendix A List of Publications **299**

Index **303**

Author Index **311**

List of figures

2.1 Word alignment from the English sentence “What’s this used for” to the Spanish sentence “para que se usa esto”. 9
2.2 Generative story of IBM-3. 9

3.1 Number of papers mentioning “neural machine translation” per year according Google Scholar. 20
3.2 Recursive autoencoder following Socher et al. (2011). The color coding indicates weight sharing. 22
3.3 The convolutional sentence model (Kalchbrenner and Blunsom, 2013). The color coding indicates weight sharing. 22
3.4 Encoder-decoder architectures with fixed-length sentence encodings. The color coding indicates weight sharing. 24
3.5 The encoder-decoder architecture of Sutskever et al. (2014). The color coding indicates weight sharing. 25
3.6 Encoder architectures. The color coding indicates weight sharing. 25
3.7 Attention weight matrix A for the translation from the English sentence “history is a great teacher .” to the German sentence “die Geschichte ist ein großer Lehrer .”. Dark shades of blue indicate high attention weights. 28
3.8 Multi-head attention with three attention heads. 28
3.9 A tensor containing a batch of three source sentences of different lengths (“the first cold shower”, “even the monkey seems to want”, “a little coat of straw” – a haiku by Basho (Basho and Reichhold, 2013)). Short sentences are padded with <pad>. The training loss and attention masks are visualized with green (enabled) and red (disabled) background. 29
3.10 The RNNsearch model following Bahdanau et al. (2015). The color coding indicates weight sharing. Gray arrows represent attention. 30
3.11 Illustration of the attention mechanism in RNNsearch (Bahdanau et al., 2015). 32

3.12 Types of 1D-convolution used in NMT. The color coding indicates weight sharing. . . . 33
3.13 NMT with a convolutional encoder and a convolutional decoder like in the ConvS2S architecture (Gehring et al., 2017b). The color coding indicates weight sharing. Gray arrows represent attention. . . . 34
3.14 Purely attention-based NMT as proposed by Vaswani et al. (2017) with two layers. The color coding indicates weight sharing. Gray arrows represent attention. . . . 35
3.15 Comparison of NMT architectures. The three inputs to attention modules are (from left to right): keys (K), values (V), and queries (Q) as in Fig. 3.8. . . . 38
3.16 Comparison between greedy (highlighted in green) and beam search (highlighted in orange) with beam size 2. . . . 40
3.17 Ensembling four NMT models. . . . 42
3.18 Distribution of words in the English portion of the English-German WMT18 training set (5.9M sentences, 140M words). . . . 49
3.19 First four merge operations of the byte pair encoding (BPE) algorithm on the text “if youth, throughout all history, had had a champion” (the opening phrase of the 260 page novel *Gadsby* from EV Wright which deliberately contains only words without the letter ‘E’). Counts indicate the frequency of symbol bigrams in the text. . . . 53
3.20 Hierarchical representation of the merge operations in the example in Fig. 3.19. 53
3.21 Performance of a Transformer model on English-German (WMT15) under varying beam sizes. The BLEU score peaks at beam size 10, but then suffers from a length ratio (hypothesis length / reference length) below 1. The log-probabilities are shown as a ratio with respect to greedy decoding. . . . 58
3.22 The length deficiency in NMT translating the English source sentence “Her husband is a former Tory councillor.” into German following Murray and Chiang (2018). The NMT model assigns a better score to the short translation “Ihr Mann ist ein ehemaliger Stadtrat.” than to the greedy translation “Ihr Mann ist ein ehemaliger Stadtrat der Tory.” even though it misses the former affiliation of the husband with the Tory Party. . . . 59
3.23 Neural networks with external memory. . . . 73

4.1 Update of the decoder hidden state in standard NMT and syntactically guided NMT. Context vectors and recursive connections are omitted in this figure for the sake of simplicity. Layers are annotated with their dimensionality in a standard RNNsearch setup. . . . 89

4.2 Greedy decoding in syntactically guided NMT. 91
4.3 NMT unconstrained and constrained search spaces over the vocabulary $\{A, B, C, D, \text{</s>}\}$. 92
4.4 Performance with NPLM over beam size on English-German *news-test2015*. A beam of 12 corresponds to row 15 in Tab. 4.5. 95
4.5 The impact of Hiero lattice size on English-German *news-test2015*. 8,510 nodes per lattice corresponds to row 14 in Tab. 4.5. 97
4.6 "Flower automata" for calculating edit distances. 98
4.7 Combining Hiero and NMT via edit distance transducer. 99
4.8 UNK extension transducer U. 99
4.9 Percentage of $\widehat{\mathbf{y}}_{Hiero}$ hypotheses found in the baseline Hiero n-best list. 103
4.10 BLEU score over the number of systems in the ensemble on *news-test2016*. . 104
4.11 Performance over n-best list size on English-German *news-test2015*. 111

5.1 Best systems on the English-German WMT *news-test2014* test set over the years (BLEU script: Moses' `multi-bleu.pl`), indicating the changes in models and experimental infrastructure. 125
5.2 Pseudo-code predictor implementations 129
5.3 Example SGNMT configuration file specifying ensembled lattice rescoring with two T2T models. 129
5.4 Multi-tokenization ensembles with the *fsttok* predictor wrapper. 130
5.5 SGNMT .ini-file for Fig. 5.4b. 131
5.6 Reduced Unified Modeling Language (UML) class diagram. 132
5.7 BLEU over the percentage of search errors. Large beam sizes yield fewer search errors but the BLEU score suffers from a length ratio below 1. 142
5.8 Even large beam sizes produce a large number of search errors. 142
5.9 Histogram over target/source length ratios. 143
5.10 Number of search errors under Beam-10 and empty global bests over the source sentence length. 144
5.11 Histogram over length ratios with minimum translation length constraint of 0.25 times the source sentence length. Experiment conducted on 73.0% of the test set. 144
5.12 Line search algorithm for SGNMT tuning. 147

6.1 Unfolding and shrinking as a combined approach to efficient ensembling. . . 152
6.2 Unfolding mimics the output of the ensemble of two single layer feedforward networks. 153
6.3 Data-free, data-bound, and SVD-based shrinking. 155

6.4 Neuron removal methods. . . . 156
6.5 SVD-based shrinking. . . . 159
6.6 Impact of shrinking on the BLEU score. . . . 162

7.1 Difference between RNNsearch and the bag2seq model. . . . 171
7.2 Building the hypothesis space for the input sentence “In a such situaction there is no other way .”. . . . 176
7.3 The edit transducer E. The σ-label can match any input symbol. . . . 177
7.4 The penalization transducer P. The ϕ-label can match any input except $<$ corr $>$ and $<$ mcorr $>$. . . . 177
7.5 Deletion FST D which can map any token in the list R from Tab. 7.7 to ε. The σ-label matches any symbol and maps it to itself. . . . 182
7.6 Insertion FST A for adding the symbols “,”, “-”, and “’s” at a cost of λ_{ins}. The σ-label matches any symbol and maps it to itself at no cost. . . . 183
7.7 Pipeline of the system combination following Yuan et al. (2019). . . . 184
7.8 Performance using back-translation on English-Turkish. Synthetic sentences are mixed at a ratio of 1:n where n is plotted on the x-axis. . . . 191
7.9 Convergence of NMT training with and without LM on English-Turkish. . . . 191
7.10 English-Turkish BLEU over training set size. . . . 194
7.11 Our modified Intra-Inter Transformer architecture with two separate attention layers. . . . 196
7.12 Intra-sentential and inter-sentential attention masks for an English example from news-test2017. Document-level context helps to predict the next word (‘vinyl’). . . . 197

8.1 The multilevel_textnorm model. . . . 206
8.2 The ffcontext_textnorm model. . . . 207
8.3 Using context vectors in the decoder network. . . . 208
8.4 Accuracy vs. speed for our different neural architectures in different sizes. . . . 209
8.5 The constrained systems are able to make better use of small training sets. . . . 212
8.6 A scatter plot of constrained and unconstrained NON_TRIVIAL error rates for our three architectures in different sizes. Constrained and unconstrained error rates are only weakly correlated. . . . 214
8.7 The translation and the alignment derived from the operation sequence in Tab. 8.7. 217
8.8 Target-side tree representation of the operation sequence in Tab. 8.7. . . . 218
8.9 AER and BLEU training curves for OSNMT on the Japanese-English dev set. 226
8.10 Encoder-decoder attention weights. . . . 227

8.11 Decoder self-attention weight matrix in layer 5, head 3; attending to the first position (constant) if the target-side write head (marked with blue) is at the end of the sentence (lines 1-5 and 17-21). 228
8.12 Decoder self-attention weight matrix in layer 5, head 6 with long-range dependencies. 229

List of tables

2.1 Semirings used in this report. $\oplus_{\log}$ is defined as $a \oplus_{\log} b := -\log(\exp(-a) + \exp(-b))$. 13

3.1 Common attention scoring functions. $v \in \mathbb{R}^{d_{\text{att}}}$, $W \in \mathbb{R}^{d_{\text{att}} \times d}$, and $U \in \mathbb{R}^{d_{\text{att}} \times d}$ in additive attention are trainable parameters with d_{att} being the dimensionality of the attention layer. 27
3.2 Types of convolution and their number of parameters. 34
3.3 Number of parameters in the original RNNsearch model (Bahdanau et al., 2015) as presented in Sec. 3.6.3 (1000 hidden units, 620-dimensional embeddings). The model size highly depends on the vocabulary size. 48
3.4 Summary of studies comparing characters and subword-units for neural machine translation. 54

4.1 Summary of studies comparing traditional statistical machine translation and neural machine translation. 87
4.2 Parallel texts and vocabulary coverage on En-De *news-test2014*. 93
4.3 Parallel texts and vocabulary coverage on En-Fr *news-test2014*. 93
4.4 BLEU scores on *news-test2014* for En-De and En-Fr. NMT-LV refers to the RNNSEARCH-LV model (Jean et al., 2015a) for large output vocabularies. . 94
4.5 BLEU English-German *news-test2015* scores calculated with `mteval-v13a.pl`. 95
4.6 Time for lattice preprocessing operations on English-German *news-test2015*. 96
4.7 English-German lower-cased BLEU scores calculated with `mteval-v13a.pl`. 102
4.8 Projection methods on *news-test2016* with NMT 8-ensemble. 103
4.9 Breakdown of the distances measured between NMT and Hiero along the shortest path in C on *news-test2016*. 104
4.10 BLEU scores on *news-test2016* for different vocabulary sizes (single NMT). Each individual NMT system is combined with Hiero as described in Sec. 4.4.2. 104

4.11 English-German lower-cased BLEU scores calculated with `mteval-v13a.pl`. **LMBR** as described in Sec. 4.5.1. . . . 110
4.12 Japanese-English cased BLEU scores calculated with Moses' `multi-bleu.pl`. **LMBR** as described in Sec. 4.5.1. . . . 110
4.13 Training data sizes for English-German and German-English after filtering. . 115
4.14 Training data sizes for Chinese-English after filtering. . . . 115
4.15 Number of model parameters. . . . 116
4.16 Impact of the effective batch size on Transformer training on en-de newstest2017 after 3,276M training tokens, beam size 4. . . . 117
4.17 Training setups for our neural models on all language pairs. . . . 118
4.18 Single architecture results on all language pairs for single systems and 2-ensembles. . . . 119
4.19 Model combination with ensembling and MBR. 'Full' indicates models in the "full-posterior group", 'MBR' in the 'MBR-based group' (Eq. 4.25). *: A system consisting of multiple 'Full' models is an ensemble. The LSTM and SliceNet models are already 2-ensembles by themselves. . . . 119
4.20 BLEU scores of the submitted systems (row 11 in Tab. 4.19). . . . 120

5.1 NMT toolkits that have been updated in the past year (as of February 2019). GitHub stars indicate the popularity of tools on GitHub. . . . 126
5.2 Predictor operations for the NMT, FST, n-gram LM, and counting modules. . 127
5.3 Currently implemented predictors as of June 2019 (SGNMT version 0.6). . . 128
5.4 Currently implemented predictor wrappers as of June 2019 (SGNMT version 0.6). . . . 131
5.5 Currently implemented decoding strategies as of June 2019 (SGNMT version 0.6). . . . 134
5.6 NMT with exact inference. In the absence of search errors, NMT often prefers the empty translation, causing a dramatic drop in length ratio and BLEU. . . . 141
5.7 *: The recurrent LSTM, the convolutional SliceNet (Kaiser et al., 2017), and the Transformer-Big systems are strong baselines from our WMT'18 shared task submission (Sec. 4.6). . . . 143
5.8 Exact search under length constraints. Experiment conducted on 48.3% of the test set. . . . 145
5.9 Length normalization fixes translation lengths, but prevents exact search from matching the BLEU score of Beam-10. Experiment conducted on 48.3% of the test set. . . . 145

5.10 BLEU scores of SGNMT with different NMT back ends on the complete KFTT test set (Neubig, 2011) computed with `multi-bleu.pl`. All neural systems are BPE-based (Sennrich et al., 2016c) with vocabulary sizes of 30K. The SMT baseline achieves 18.1 BLEU. 149

6.1 Shrinking layers of the unfolded network on Ja-En to their original size. . . . 160
6.2 Compensating for neuron removal in the data-bound algorithm. Row (d) corresponds to row (f) in Tab. 6.1. 161
6.3 Time measurements on Ja-En. Layers are shrunk to their size in the original NMT model. 162
6.4 Layer sizes of our setups for Ja-En. 163
6.5 Our best models on Ja-En. 164
6.6 Our best models on En-De. 164

7.1 BLEU scores for PTB word-ordering for different search heuristics with beam=512. NGRAM-512, LSTM-512 are quoted from Schmaltz et al. (2016), and +GW indicates Gigaword data. The best results for a given model and set are highlighted in bold. 173
7.2 Impact of search heuristics and beam sizes under different model combinations (test) on PTB data. The best results for a given setup are highlighted in bold. . 174
7.3 Results without using annotated training data. Systems are tuned with respect to the metric highlighted in gray. Input lattices I are derived from the source sentence as in Fig. 7.2a. 179
7.4 Results with using annotated training data. Systems are tuned with respect to the metric highlighted in gray. Input lattices I are derived from the Moses 1000-best list as in Fig. 7.2c. Row 3 is the SMT baseline. 180
7.5 Improvements over SMT baselines. 181
7.6 Oracle error rates for different hypothesis spaces using the first annotator in CoNLL-2014. 181
7.7 List of tokens R that can be deleted by the deletion transducer D in Fig. 7.5. . 182
7.8 Results on the low-resource track. The λ-parameters are tuned on the BEA-2019 dev set. 183
7.9 Number of correction types in CoNLL-2014 and BEA-2019 Dev references. . 184
7.10 ERRANT scores on BEA-2019 Dev for the components in Fig. 7.7. 184
7.11 Parallel training data. 188
7.12 Monolingual training data. 188
7.13 Summary of NMT settings for all models in this section. 189

7.14 Comparison of our PRENORM and POSTNORM combination strategies with shallow fusion (Gulcehre et al., 2015) and cold fusion (Sriram et al., 2018) under a recurrent neural network language model (RNN-LM). 190
7.15 Comparison between using a recurrent LM (RNN) and an n-gram based feed-forward LM (FFN) on English-Turkish. 192
7.16 Combining an RNN-LM and a feedforward LM with the translation model using the POSTNORM strategy. 192
7.17 Perplexity and average entropies of the distributions generated by our systems on the English-Turkish dev set. 193
7.18 BLEU n-gram precisions for Estonian-English. 193
7.19 Translation samples from the Estonian-English test set. 194
7.20 Language model perplexities of different neural language models. The standard model has 448M parameters, Intra-Inter has 549M parameters. 198
7.21 Using different kinds of language models for translation on `news-test2018`. The Big models are fine-tuned on old WMT test sets with EWC as described by Stahlberg et al. (2019b). The PBSMT baseline gets 26.7 BLEU on English-German and 27.5 BLEU on German-English. 198
7.22 English-German and German-English primary submissions to the WMT19 shared task. 199
7.23 Comparison of our English-German system with the winning submissions over the past two years. 200

8.1 Syntactic representations of the sentence "No complications occurred" following Saunders et al. (2018). 202
8.2 Japanese-English syntax-based NMT with Transformer based models (Saunders et al., 2018). The first three rows are single models, the last five rows are multi-representation ensembles. 203
8.3 Error rates and decoding speeds of our neural text normalization models. ALL error rates are computed on all tokens, NON_TRIVIAL is restricted to non-trivial samples (i.e. no < `self` > token). 209
8.4 Impact of the context window size on the NON_TRIVIAL error rates for English. We use the `concat` context strategy for `multilevel_textnorm` and the `init` strategy for `ffcontext_textnorm`. 210
8.5 Using different tokenization strategies for $\text{tok}_{\text{context}}(\cdot)$ (rows) and $\text{tok}_{\text{mid}}(\cdot)$ (columns) in the `multilevel_textnorm` (`concat`) model for English. 211
8.6 Constraining English neural text normalization models with covering grammars. 211

8.7 Generation of the target sentence “stable operation of 2000 hr was confirmed”. The neural model produces the linear sequence of operations in the first column. The positions of the source-side read head and the target-side write head are highlighted. The marker in the target sentence produced by the i-th SET_MARKER operation is denoted with ‘X_{i+1}’; X_1 is the initial marker. We denote INSERT(t) operations as t to simplify notation. 217
8.8 Derivation in $\mathscr{G}$ for the example of Tab. 8.7. 219
8.9 Training set sizes. 222
8.10 Generating training alignments on the subword level. 223
8.11 Frequency of invalid OSNMT sequences produced by an unconstrained decoder on the ja-en test set. 224
8.12 Comparison between plain text and OSNMT on Spanish-English (es-en), Portuguese-English (pt-en), and Japanese-English (ja-en). 224
8.13 Examples of Portuguese-English translations together with their (subword-) alignments induced by the operation sequence. Alignment links from source words consisting of multiple subwords were mapped to the final subword in the training data, visible for ‘temperamento’ and ‘pennisetum’. 225
8.14 Comparison between OSNMT and using the attention matrix from forced decoding with a recurrent model. 226

Nomenclature

Roman Symbols

I	Source sentence length
J	Target sentence length
$\mathbf{x}$	Source language sentence
$\mathbf{y}$	Target language sentence

Greek Symbols

$\mathbb{G}_n$	Set of first n natural numbers
λ	Weight vector
Θ	Model parameters
Σ	Vocabulary
Σ_{src}	Source language vocabulary
Σ_{trg}	Target language vocabulary

Subscripts

i	Index of a token in the source sentence
j	Index of a token in the target sentence

Other Symbols

<s>	Begin-of-sentence symbol
</s>	End-of-sentence symbol

<pad> Padding symbol

Acronyms / Abbreviations

BiRNN Bidirectional recurrent neural network

BLEU Score for measuring translation quality (bilingual evaluation understudy)

BPE Byte pair encoding

CKY Cocke–Kasami-Younger (algorithm)

CPU Central processing unit

DFS Depth-first search

FSA Finite-state automaton

FSM Finite-state machine

FST Finite-state transducer

GAN Generative adversarial network

GEC Grammatical error correction

GNMT Google's neural machine translation

GPU Graphics processing unit

GRU Gated recurrent unit

Hiero A hierarchical phrase-based model for statistical machine translation

HiFST Cambridge's hierarchical phrase-based statistical machine translation system based on OpenFST

LMBR Lattice minimum Bayes-risk decoding

LM Language model

LSTM Long short-term memory network

MBR Minimum Bayes-risk decoding

MERT Minimum error rate training

MLE Maximum likelihood estimation

MTG Multitext grammar

MT Machine translation

NCE Noise-contrastive estimation

NLP Natural language processing

NMT Neural machine translation

OSNMT Operation sequence neural machine translation

RCTM Recurrent continuous translation model

RNMT+ RNN-based NMT model plus

RNN Recurrent neural network

RTN Recursive transition network

SCFG Synchronous context-free grammar

SGD Stochastic gradient descent

SGNMT Syntactically guided neural machine translation (toolkit)

SMT Statistical machine translation

SVD Singular value decomposition

TM Translation model

UNK Unknown word

WFSA Weighted finite-state automaton

WFST Weighted finite-state transducer

WMT Before 2016: Workshop on Statistical Machine Translation. After 2016: Conference on Machine Translation

Wherever you go becomes a part of you somehow.

Anita Desai

1 Introduction

1.1 Motivation

Deep learning has revolutionized virtually every aspect of machine learning (Goldberg, 2016). Recent advances in designing and training large-scale neural networks are the main reason why artificial intelligence has become the mantra of our age. One of the pillars of the connectionist paradigm is end-to-end training: models learn to predict from the raw data without any intermediary steps like preprocessing or feature engineering. This approach has been taken to the extreme, perhaps most famously by WaveNet (van den Oord et al., 2016a), a neural Text-to-Speech system which directly generates raw waveforms, and Translatotron (Jia et al., 2019b), Google's speech-to-speech translation system that directly transforms waveforms in one language to waveforms in another language without any intermediate textual representation. End-to-end training can be seen in a wider context as yet another milestone of a general shift in our field which is best characterized by a well-known quote from the late Fred Jelinek:

> Every time I fire a linguist, the performance of our speech recognition system goes up.
> — Fred Jelinek

Modern machine learning algorithms claim to require minimal human involvement in system design (Graves and Jaitly, 2014). Automatically learned features have been shown to outperform

highly engineered and hand-crafted features in almost all areas of natural language processing (NLP). A prime example of this development is the field of machine translation (MT), the automatic translation of written text from one natural language such as German into another natural language such as English. Research in MT has shifted from rule-based MT which often used hand-crafted rules, to phrase-based statistical MT (SMT) which learns from bilingual text but uses a large number of different hand-engineered features, and finally to neural MT (NMT) which tackles translation with a single neural network.

Deep learning and end-to-end training have certainly pushed the overall accuracy of models to new limits. Claims of near or complete human parity in language translation are becoming more frequent (Hassan et al., 2018; SDL, 2018; Wu et al., 2016b), although they often do not stand up to closer scrutiny (Läubli et al., 2018; Li and Chen, 2019; Schwarzenberg et al., 2019; Tomasello, 2019; Toral et al., 2018). However, there are a number of issues which still seem to be deeply intertwined with the new paradigm (Sculley et al., 2018).

First, systems like WaveNet are difficult to adapt in practice by other researchers with fewer computation resources or training data. Training Google's Neural Machine Translation system for a single language pair (English-French) takes 9 days on 96 NVIDIA K80 GPUs (Wu et al., 2016b). This impedes scientific progress as only few groups in the world have access to this amount of computational resources. Second, neural models are hard to interpret since information in these networks is represented by real-valued vectors or matrices (Ding et al., 2017). Explainable and interpretable deep learning is still an open research question (Doshi-Velez and Kim, 2017; Lipton, 2018; Montavon et al., 2018; Ribeiro et al., 2016), particularly in the context of natural language processing (Alvarez-Melis and Jaakkola, 2017; Ding et al., 2017; Karpathy et al., 2015; Li et al., 2016a). Related to the interpretability problem is the fact that neural models often cannot give guarantees about their predictions, and (very) occasionally produce unacceptable output. Examples of this deficiency are neural machine translation models generating non-words which are not part of any human language, or neural text normalization systems with 'catastrophic' errors such as reading '11/10/2008' as 'the tenth of october two thousand eight' rather than 'the tenth of november two thousand eight' (Sproat and Jaitly, 2016). These kind of errors do not affect the overall accuracy greatly as they do not occur very often, but they do impair the usefulness of the system for the end user.

In this thesis, we will diverge from the mainstream end-to-end scheme. We will present several instances in which purely neural models benefit from treating the task not just as raw data mapping problem. This benefit can be a boost in accuracy or fluency, the prevention of 'catastrophic' errors, improved interpretability, or a theoretical connection between neural models and formal grammars. We mainly focus on machine translation, but also investigate other areas of NLP that involve sequence models such as text normalization, grammatical error

correction, and language generation. The next section will give a more concrete overview of our contributions in these fields.

1.2 Contributions

A summary of the original contributions of this thesis is as follows:

- We show that the structured search spaces defined by syntactic SMT approaches such as Hiero (Chiang, 2007) can be used to guide NMT, and that rescoring SMT lattices with NMT can yield gains over both baselines (Stahlberg et al., 2016b).
- However, combining NMT and SMT via rescoring is often too constraining for very strong neural models. Therefore, we propose an edit-distance-based loose coupling scheme using finite state transducers (FSTs). We have used this scheme in a successful submission to the WMT16 evaluation campaign (Stahlberg et al., 2016a).
- We also present a novel scheme for NMT-SMT hybrids based on a minimum Bayes risk (MBR) formulation (Stahlberg et al., 2017a). Our MBR-based framework has been adopted by the industry (Iglesias et al., 2018). We extended our framework to multiple models in our submission to the WMT18 MT competition (Stahlberg et al., 2018b), ranking second in all our three language pairs in terms of human assessment.
- We develop a software package called SGNMT for machine translation research (Stahlberg et al., 2017b, 2018d). SGNMT is highly versatile and has been used for most of the research work done by the Cambridge MT group and for a number of fourth year and MPhil projects. SGNMT is also used for teaching as part of three coursework practicals about recasing, NMT decoding strategies, and grammatical error correction.
- We use our SGNMT decoder to analyze model errors and search errors in NMT (Stahlberg and Byrne, 2019b), revealing that the model often assigns the highest probability to the empty translation. In practice, NMT relies on a large number of search errors to prevent the decoder from finding the empty translation.
- Ensembling (averaging the predictions of multiple models) improves NMT quality but is cumbersome and slow. We show that an ensemble can be unfolded into a single large neural network which imitates the output of the ensemble system. We also describe a set of techniques to shrink the unfolded network by reducing the dimensionality of layers. The resulting network has the size and decoding speed of a single NMT network but performs on the level of a 3-ensemble system (Stahlberg and Byrne, 2017).

- We show how a combination of neural and symbolic count-based language models (LMs) works best for the task of reordering a bag of words back to the original sentence order (Hasler et al., 2017b). Cross-lingual reordering is one of the major challenges in MT, and studying word reordering as isolated problem helps to understand the limitations of current approaches.

- We also describe a novel way to use language models in NMT training (Stahlberg et al., 2018a). This is particularly useful for MT as monolingual data is usually much more abundant than bilingual data. We combine the scores of a pre-trained and fixed language model with the scores of a translation model while the translation model is trained from scratch. To achieve that, we train the translation model to predict the residual probability of the training data added to the prediction of the LM. This enables NMT to focus its model capacity on modeling the source sentence since it can rely on the LM for fluency.

- Furthermore, we find that FSTs are a very effective way to define the search space for neural grammatical error correction through spell checkers, morphology databases, and potentially SMT systems (Stahlberg et al., 2019a). We show how to design state-of-the-art neural error correction systems by constraining a neural decoder with these FSTs.

- We also explore strategies for incorporating target syntax into NMT (Saunders et al., 2018). We report state-of-the-art results on a difficult Japanese-English test set by using multiple syntax representations and a training schedule based on delayed SGD updates.

- Another use of hierarchical models are formal covering grammars in text normalization. We investigate novel neural architectures which frame text normalization as contextual sequence-to-sequence problem, and show how covering grammars are effective in preventing 'catastrophic' errors (Zhang et al., 2019a).

- Finally, we present our work on operation sequence neural machine translation (OSNMT). OSNMT achieves explainable NMT by changing the output representation to explain itself (Stahlberg et al., 2018c). The translation process is represented by a linear sequence of operations which represent both word reordering and lexical translation. The operation sequences can be used to derive word alignments for explaining each output token with a link into the source sentence. OSNMT also has strong theoretical connections to hierarchical SMT as an OSNMT sequence can be seen as a parse through a formal multitext grammar.

1.3 Organization of this Thesis

The first part of this thesis (Chapters 2-3) contains relevant related work, whereas the second part (Chapters 4-8) is devoted to own work. Ch. 2 introduces weighted finite state transducers and statistical machine translation (SMT) briefly. Ch. 3 contains an introduction to neural machine translation (NMT) and a comprehensive overview of current research directions in this area. Our work on SMT-NMT hybrids is presented in Ch. 4, followed by an implementation-focused Ch. 5. Our work on efficient ensembling by unfolding and shrinking is discussed in Ch. 6. Ch. 7 presents our investigations into using language models for neural sequence modelling, both in training and decoding. We explore various ways to use hierarchical models in neural sequence models in Ch. 8. We finish with our conclusion in Ch. 9.

Parts of this thesis have been published in conference proceedings and journal articles. A publication list is provided in Appendix A.

1.4 Notations

Throughout this thesis we will denote the source sentence of length I as $\mathbf{x}$. We use the subscript i to index tokens in the source sentence. We refer to the source language vocabulary as Σ_{src}.

$$\mathbf{x} = x_1^I = (x_1, \ldots, x_I) \in \Sigma_{src}^I \tag{1.1}$$

The translation of source sentence $\mathbf{x}$ into the target language is denoted as $\mathbf{y}$. We use an analogous nomenclature on the target side.

$$\mathbf{y} = y_1^J = (y_1, \ldots, y_J) \in \Sigma_{trg}^J \tag{1.2}$$

In case we deal with only one language we drop the subscript src. For convenience we represent tokens as indices in a list of subwords or word surface forms. Therefore, Σ_{src} and Σ_{trg} are the first n natural numbers:

$$\Sigma = \mathbb{G}_n = \{n' \in \mathbb{N} | n' \leq n\} \tag{1.3}$$

where $n = |\Sigma|$ is the vocabulary size. Additionally, we use the projection function π_k which maps a tuple or vector to its k-th entry:

$$\pi_k(z_1, \ldots, z_k, \ldots, z_n) = z_k. \tag{1.4}$$

For a matrix $A \in \mathbb{R}^{m \times n}$ we denote the element in the p-th row and the q-th column as $A_{p,q}$, the p-th row vector as $A_{p,:} \in \mathbb{R}^n$ and the q-th column vector as $A_{:,q} \in \mathbb{R}^m$. For a series of m

n-dimensional vectors $a_p \in \mathbb{R}^n$ ($p \in [1,m]$) we denote the $m \times n$ matrix which results from stacking the vectors horizontally as $(a_p)_{p=1:m}$ as illustrated with the following tautology:

$$A = (A_{p,:})_{p=1:m} = ((A_{:,q})_{q=1:n})^T. \tag{1.5}$$

I have worried a good deal about the probable naivete of the ideas here presented; but the subject seems to me so important that I am willing to expose my ignorance, hoping that it will be slightly shielded by my intentions.

Warren Weaver (Translation, 1949)

2

Statistical Machine Translation and Symbolic Models

2.1 Motivation

Statistical machine translation (SMT) had been the *de facto* standard for open domain machine translation for decades. The core of many classical SMT systems are *count-based* probability models in which we estimate the probabilities of words or phrases based on counts in a training corpus. In recent years, neural machine translation (NMT) which will be presented in Ch. 3 has largely superseded SMT as the prevalent approach to MT both in research and industry. However, as we argue in Ch. 4, NMT and SMT have complementary strengths and weaknesses and differ markedly in how they define probability distributions over translations and what search procedures they use. A large portion of this thesis is therefore devoted to exploring the potential of SMT-NMT hybrids. This chapter provides a brief introduction to SMT without going into details on the topic. The chapter is rather intended to give the reader just enough insight for following our work on hybrid systems. We largely follow Koehn (2010) and Chiang (2007) with minor changes in nomenclature.

2.2 N-gram Language Models

Language models (LMs) are probabilistic models of human language. That means that they assign a probability $P(\mathbf{y})$ to a sentence $\mathbf{y}$ which captures how likely the word sequence $\mathbf{y}$ is in that language. Of course, these probabilities are highly context-dependent: the sentence *The May Balls take place mid June.* is generally rather unlikely unless the context is Cambridge. Therefore, LMs are of little use by their own and just reflect prior beliefs about a language. The LM must be paired with other models, such as acoustic models (in speech recognition), optical models (in optical character recognition) or translation models (in machine translation). In statistical machine translation, LMs are often crucial to improve translation quality (Koehn, 2010, Ch. 7).

A very widely used class of language models are n-gram LMs. For example, a trigram language model ($n = 3$) has the following form:

$$P(\mathbf{y}) \overset{\text{Def.}}{=} P(y_1^J) \overset{\text{Chain rule}}{=} \prod_{j=1}^{J} P(y_j|y_1^{j-1}) \approx P(y_1) \cdot P(y_2|y_1) \cdot \prod_{j=3}^{J} P(y_j|y_{j-2}, y_{j-1}). \tag{2.1}$$

We first factorize $P(\mathbf{y})$ using the chain rule, and then approximate $P(y_j|y_1^{j-1})$ (conditioned on the entire history) with $P(y_j|y_{j-2}, y_{j-1})$ (conditioned only on the previous two words) by making a second order Markov assumption. The simplified probability can be modelled by count-based models (Brown et al., 1990; Heafield et al., 2013) or feedforward neural networks (Bengio et al., 2003, 2006; Schwenk et al., 2006; Vaswani et al., 2013). There are numerous reasons for the popularity of n-gram language models: they are easy to train and implement, are robust, often simplify decoding because of the bounded history length, and work well in practice.

2.3 Word-based Models

Early statistical models for machine translation were heavily inspired by the noisy-channel model. In this model, a sentence $\mathbf{y}$ is sent through a noisy channel which disturbs it in such a way that a sentence $\mathbf{x}$ in another language comes out. Since the channel is noisy, this mapping is not deterministic but better described by a distribution $P(\mathbf{x}|\mathbf{y})$ (the translation model). The translation model can be combined with the LM using Bayes rule (Koehn, 2010, Eq. 4.23):

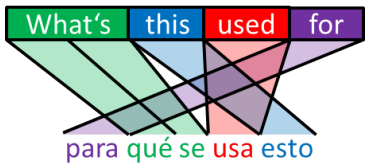


Fig. 2.1 Word alignment from the English sentence "What's this used for" to the Spanish sentence "para que se usa esto".

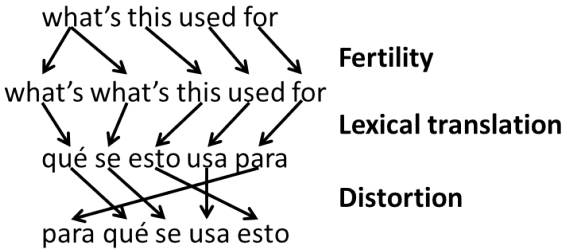


Fig. 2.2 Generative story of IBM-3.

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}) = \arg\max_{\mathbf{y}} \frac{P(\mathbf{x}|\mathbf{y}) \cdot P(\mathbf{y})}{P(\mathbf{x})} = \arg\max_{\mathbf{y}} \underbrace{P(\mathbf{x}|\mathbf{y})}_{\text{Translation model}} \cdot \underbrace{P(\mathbf{y})}_{\text{Language model}} . \quad (2.2)$$

The early word-based generative models for $P(\mathbf{x}|\mathbf{y})$ are described by a *generative story* which defines how sentence $\mathbf{y}$ is transformed by the channel to sentence $\mathbf{x}$ in the other language. For example, the story in the IBM models 1-5 (Brown et al., 1993) claims that the words in $\mathbf{y}$ are multiplied, translated separately, and then scrambled around to form $\mathbf{x}$ (Knight, 1999). The probabilities of these steps are modelled by increasingly sophisticated count-based models.

2.4 Word Alignments

The generative stories of the IBM models convey the notion that each word in the generated sentence $\mathbf{x}$ is generated by a word in the original sentence $\mathbf{y}$.[1] A convenient way to represent such word-level relationships are *word alignments*. A word alignment links words in the source sentence and words in the target sentence which are translations of each other (Fig. 2.1). Many alignment models assume a 1:n relationship such that alignments can be formalized via an

[1] Note that although we normally denote the source sentence as $\mathbf{x}$ and the target sentence as $\mathbf{y}$ as described in Sec. 1.4, the roles are switched here since IBM models were originally applied in the noisy-channel model (Eq. 2.2).

alignment function $a : \mathbb{G}_I \rightarrow \mathbb{G}_J$.[2] The simplest model in the IBM model hierarchy (IBM-1) which is solely based on lexical translation probabilities can be described as (Brown et al., 1993, Eq. 5):

$$P(\mathbf{x}|\mathbf{y}, a) \propto J^{-I} \prod_{i=1}^{I} P(x_i|y_{a(i)}). \tag{2.3}$$

$P(x|y)$ is represented as a large translation table that associates word translations with probabilities. Higher order IBM models such as IBM-3 (Fig. 2.2) do not only use lexical translation tables but also model fertility (the number of words a source word generates) and word reorderings in their generative story.

Word alignments are not only fundamental for defining and training IBM models, but they also play a central role in phrase-based machine translation (Sec. 2.5). Besides their technical necessity in SMT, word alignments are very useful for practical machine translation as they can be used for enforcing terminology constraints (Hasler et al., 2018), preserving text formatting, correcting translation errors in post-processing, or incorporating user feedback by displaying translation options for specific words. However, although often superior to SMT in terms of translation quality, neural machine translation does not rely on word alignments and is thus problematic with respect to these practical considerations. We will present a method in Sec. 8.4 which aims to reintroduce the concept of word alignments to neural MT.

2.5 Phrase-based Translation

A major drawback of word-based models is that they do not explicitly model lexical word context, and thus often break down when multiple words are to be translated to a single word. *Phrase-based* models quickly emerged to fix this shortcoming of word-based models. Phrase-based MT is motivated by the realization that phrases consisting of multiple words are often more reasonable units of translation. A phrase-based model segments $\mathbf{x}$ into K phrases $\bar{x}_k$ and models $P(\mathbf{x}|\mathbf{y})$ as product of phrase translation probabilities $\phi(\bar{x}_k|\bar{y}_k)$ and a basic distortion model $d(\cdot)$ (Koehn, 2010, Eq. 5.2).

$$P(\mathbf{x}|\mathbf{y}) = P(\bar{x}_1^K|\bar{y}_1^K) = \prod_{k=1}^{K} \phi(\bar{x}_k|\bar{y}_k) \cdot d(\text{start}_k - \text{end}_{k-1} - 1) \tag{2.4}$$

Therefore, phrase-based models do not use words as basic translation units but allow whole phrases to be translated directly. Phrase-based SMT was the state-of-the-art in open domain

[2] There are obvious examples in which the 1:n assumption is not valid such as translation from morphologically poor languages, multi-word idioms or spurious source words which do not have any correspondence in the target sentence. These problems can be (partially) addressed with NULL alignment and symmetrization (Och and Ney, 2003).

MT for a long time. Eq. 2.2 can be rewritten for the phrase-based model as (Koehn, 2010, Sec. 5.3.1):

$$\begin{aligned}\hat{\mathbf{y}} &= \arg\max_{\mathbf{y}} \underbrace{\prod_{k=1}^{K} \phi(\bar{x}_k|\bar{y}_k)}_{\text{Translation model}} \cdot \underbrace{\prod_{k=1}^{K} d(\text{start}_k - \text{end}_{k-1} - 1)}_{\text{Distortion model}} \cdot \underbrace{P(\mathbf{y})}_{\text{Language model}} \\ &= \arg\max_{\mathbf{y}} \underbrace{\log \prod_{k=1}^{K} \phi(\bar{x}_k|\bar{y}_k)}_{:=f_1} + \underbrace{\log \prod_{k=1}^{K} d(\text{start}_k - \text{end}_{k-1} - 1)}_{:=f_2} + \underbrace{\log P(\mathbf{y})}_{:=f_3} \\ &= \arg\max_{\mathbf{y}} f_1 + f_2 + f_3.\end{aligned} \tag{2.5}$$

Due to practical reasons, the feature functions f_l are normally scaled by weight factors λ_l (Koehn, 2010, following Eq. 5.7):

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} \sum_l \lambda_l \cdot f_l. \tag{2.6}$$

Eq. 2.6 shows that we can treat the search problem in phrase-based SMT as optimizing a weighted sum of features f_l (Och and Ney, 2002). This form is often referred to as *log-linear model combination* since we linearly combine log-likelihoods (e.g. $f_3 = \log P(\mathbf{y})$). It becomes obvious that Eq. 2.6 does not only allow us to use translation, distortion, and language models but any kind of features like word and phrase penalties or inverse translation probabilities (Koehn, 2010). The feature f_l can also by itself be a vector in which case $\lambda_l \cdot f_l$ is a dot-product.

However, the transition to phrase-based MT does not help to solve the decoding problem: how can we efficiently search for the best translation $\hat{\mathbf{y}}$, how can we implement the arg max algorithmically? Sec. 2.7 presents hierarchical machine translation which has a very elegant and rigorous way to define the search space.

2.6 Weighted Finite State Transducers

Finite state machines (FSMs) are a central concept of computer science. In automaton theory they represent the class of machines which recognize regular languages and are therefore widely used in many computer science disciplines such as programming language design, network protocols, hardware design, and natural language processing (NLP). A subclass of FSMs which is particularly relevant to NLP are weighted finite state transducers (WFSTs). WFSTs are well-studied in the NLP literature for decades and libraries exist which support

WFST operations very efficiently (Allauzen et al., 2007; Chen et al., 2018e). There is a variety of work which connects machine translation with WFSTs (Allauzen et al., 2014; Bangalore and Riccardi, 2001; de Gispert et al., 2010; Iglesias et al., 2009; Kumar and Byrne, 2005, among others), and state-of-the-art systems in other areas of NLP like speech recognition (Mohri et al., 2008; Povey et al., 2011) or optical character recognition (Bluche et al., 2014; Stahlberg and Vogel, 2015) often make extensive use of WFSTs.

Many methods which are proposed in this thesis rely heavily on WFSTs such as all three SMT-NMT hybrid techniques in Ch. 4, our SGNMT software package in Ch. 5, the grammatical error correction system described in Sec. 7.3, our framework in Sec. 8.2 for multi-representation ensembles for target-side syntax in NMT, and the covering grammar approach to neural text normalization discussed in Sec. 8.3. In this section we will introduce WFSTs formally and define useful operations on them which are used throughout this thesis.

2.6.1 Formalisms

Our formalisms generally follow Mohri (2003) with some modifications in spirit of our notations introduced in Sec. 1.4. We define a weighted transducer T as a 6-tuple $(V, s, F, \Sigma_1, \Sigma_2, E)$ where (Mohri, 2003, cf. Def. 8):

- V is the set of states,
- $s \in V$ is the unique initial state,
- $F \subset V$ is the set of final states,
- Σ_1 and Σ_2 are the input and output alphabets,
- $E \subset (\{\varepsilon\} \cup \Sigma_1) \times (\{\varepsilon\} \cup \Sigma_2) \times \mathbb{K} \times V \times V$ is the set of arcs. We write E_v for the set of all outgoing edges from a state $v \in V$:

$$E_v := E \cap ((\{\varepsilon\} \cup \Sigma_1) \times (\{\varepsilon\} \cup \Sigma_2) \times \mathbb{K} \times \{v\} \times V). \tag{2.7}$$

A transducer is usually defined over a *semiring* $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$ with an 'addition' operator denoted with '$\oplus$' and a 'multiplication' operator denoted with '$\otimes$' and neutral elements $\bar{0}$ and $\bar{1}$. In contrast to rings, elements in semirings may lack an inverse for the addition. Tab. 2.1 shows the two semirings relevant to this work.

For convenience, we will use the projection function $\pi_k(\cdot)$ which maps a tuple to its k-th entry (Eq. 1.4). An edge $e \in E$ represents an arc from state $\pi_4(e)$ to state $\pi_5(e)$ with input label $\pi_1(e)$, output label $\pi_2(e)$, and weight $\pi_3(e)$. To improve readability we adopt the notation

Semiring	$\mathbb{K}$	$\oplus$	$\otimes$	$\bar{0}$	$\bar{1}$
Log	$\mathbb{R}\cup\{\infty\}$	$\oplus_{\log}$	$+$	∞	0
Tropical	$\mathbb{R}\cup\{\infty\}$	min	$+$	∞	0

Table 2.1 Semirings used in this report. $\oplus_{\log}$ is defined as $a \oplus_{\log} b := -\log(\exp(-a)+\exp(-b))$.

from Mohri (2003) and denote the previous state as $p[e] = \pi_4(e)$, the next state as $n[e] = \pi_5(e)$, the input and output labels as $i[e] = \pi_1(e)$ and $o[e] = \pi_2(e)$, and the arc weight as $\omega[e] = \pi_3(e)$. We extend $i[\cdot]$, $o[\cdot]$, and $\omega[\cdot]$ to the domain E^* of sequences over edges:

$$\begin{aligned} i[\varepsilon] &= \varepsilon, \quad \forall p \in E^*, e \in E : i[p\cdot e] &&= i[p]\cdot i[e] \\ o[\varepsilon] &= \varepsilon, \quad \forall p \in E^*, e \in E : o[p\cdot e] &&= o[p]\cdot o[e] \\ \omega[\varepsilon] &= \bar{1}, \qquad \forall p \in E^+ : \omega[p] &&= \bigotimes_{e\in p} \omega[e] \end{aligned} \tag{2.8}$$

The set $\mathscr{P}(T)$ of *complete paths* in T is the set of paths from the initial state s to a state in F[3].

$$\mathscr{P}(T) = \{(e_1,\ldots,e_n) \in E^+ | e_1 \in E_s \wedge n[e_n] \in F \wedge \forall j \in [2,n] : n[e_{j-1}] = p[e_j]\} \tag{2.9}$$

We extend $\mathscr{P}(\cdot)$ in the following way to restrict the set based on the input and output labels:

$$\mathscr{P}'(T,w_1,w_2) = \{p \in \mathscr{P}(T) | i[p] = w_1 \wedge o[p] = w_2\} \tag{2.10}$$

The transducer T *transduces* a sequence $w_1 \in \Sigma_1^*$ to a sequence $w_2 \in \Sigma_2^*$ iff. there is a path in T with w_1 on the input labels and w_2 on the output labels (i.e. if $\mathscr{P}'(T,w_1,w_2) \neq \emptyset$). A transducer T is *regulated* if for each pair $(w_1,w_2) \in \Sigma_1^* \times \Sigma_2^*$ the following term is well-defined (Mohri, 2003, Eq. 11):

$$[[T]](w_1,w_2) = \begin{cases} \bigoplus_{p\in\mathscr{P}'(T,w_1,w_2)} \omega(p) & \text{if } \mathscr{P}'(T,w_1,w_2) \neq \emptyset \\ \bar{0} & \text{if } \mathscr{P}'(T,w_1,w_2) = \emptyset \end{cases} \tag{2.11}$$

For example, WFSTs are regulated if they contain no ε-loops (Mohri, 2003) because $\mathscr{P}'(T,w_1,w_2)$ is always finite in that case. In the remainder we consider only regulated WFSTs.

[3] If $s \in F$ we add ε to $\mathscr{P}(T)$.

Sequential transducers (Eq. 2.12) are WFSTs in which two different outgoing arcs from any given state do not share the same input label (Mohri, 1997):

$$\forall v \in V : \forall e_1, e_2 \in E_v : (i[e_1] = i[e_2]) \implies (e_1 = e_2). \tag{2.12}$$

Any WFST over the log or tropical semiring can be turned into an equivalent sequential transducer using the operations `Determinize`, `RmEpsilon`, and optionally `Minimize` (Mohri, 1997).

If all input labels are equal to the corresponding output labels ($i[e] = o[e]$ for all $e \in E$), we call the WFST a *weighted finite state automaton* (WFSA). In this case, we introduce a short-hand notation for sequences which are accepted by the WFSA:

$$w \in T :\Longleftrightarrow [[T]](w,w) \neq \bar{0} \tag{2.13}$$

For WFSAs we denote the arc label as $l[e]$ to make clear that there is only one such label (i.e. $l[e] = i[e] = o[e]$ for all $e \in E$).

2.6.2 Operations on WFSTs

Besides `Determinize`, `RmEpsilon`, and `Minimize`, we introduce some more WFST operations which we will use throughout this thesis.

Composition We will require *composition* as tool for building complex automata from simpler ones. The composition of two weighted transducers T_1, T_2 (denoted as $T_1 \circ T_2$) is defined as (Mohri, 2003, Eq. 12):

$$[[T_1 \circ T_2]](w_1, w_2) = \bigoplus_z [[T_1]](w_1, z) \otimes [[T_2]](z, w_2). \tag{2.14}$$

Intuitively, $T_1 \circ T_2$ transduces w_1 to w_2 if there is a sequence z for which T_1 transduces w_1 to z and T_2 transforms z to w_2. Composition of WFSTs is related to the mathematical composition of functions: if both T_1 and T_2 are input and output deterministic and we ignore weights, we can view them as functions which map an input sequence to an output sequence. In that case, $T_1 \circ T_2$ corresponds to the composition of these functions. However, note that the composition operator on functions normally defines the order of the operands differently.

Pushing Another useful WFST operation is `Push` (Allauzen et al., 2007; Mohri, 1997; Mohri and Riley, 2001). The `Push` operation towards the initial state transforms a transducer $T = (V, s, F, \Sigma_1, \Sigma_2, E)$ to an equivalent machine $T' = (V, s, F, \Sigma_1, \Sigma_2, E')$ for which holds:

$$\forall v \in V \setminus \{s\} : \bigoplus_{e \in E'_v} \omega(e) = \bar{1}. \tag{2.15}$$

For each non-initial state in T' the weights on outgoing transitions sum up to $\bar{1}$. Transducers with this property are called *stochastic* as we can interpret the outgoing weights as probability distribution over labels.

Projection Projection converts a finite state transducer to a finite state automaton by replacing all output labels by the corresponding input labels ('input projection' denoted by Π_{input}) or vice versa ('output projection' denoted by Π_{output}).

2.7 Hierarchical Phrase-based Machine Translation

Hierarchical phrase-based machine translation systems such as Hiero (Chiang, 2005, 2007) build on phrase-based translation. Instead of representing sentences as a flat sequence of words, hierarchical phrase-based MT represents them with tree structures which are able to describe 'syntactic relationships between words and phrases' (Koehn, 2010). However, it is important to note that syntax in hierarchical MT often refers to the concept of formal grammars in computer science, and does not necessarily correspond to the sentence syntax in a linguistic sense. The underlying formalism is based on weighted synchronous context-free grammars (Aho and Ullman, 1969; Lewis II and Stearns, 1968) (SCFGs).

2.7.1 Synchronous Grammars

This section roughly follows the formalism used by Chiang (2005, 2007) rather than the original one from Lewis II and Stearns (1968). An SCFG is defined by two disjunct alphabets of terminals (T) and non-terminals (NT), a start symbol $S \in NT$, and a set of rewrite rules R of the following form (Chiang, 2005, Eq. 9):

$$X \to \langle \gamma, \alpha, \sim \rangle \tag{2.16}$$

where $X \in NT$ is a single non-terminal, $\gamma, \alpha \in (NT \cup T)^*$ strings of terminal and non-terminal symbols, and $\sim$ a bijection between the positions of non-terminals in γ and α. An SCFG can be used to represent a bilingual pair of source sentence $\mathbf{x}$ and target sentence $\mathbf{y}$ with a hierarchical structure. Suppose there is a derivation $D \in R^+$ which yields both $\mathbf{x}$ and $\mathbf{y}$:

$$\langle S, S \rangle \Rightarrow_D \langle \mathbf{x}, \mathbf{y} \rangle \tag{2.17}$$

where '$\Rightarrow_D$' is the transitive completion of '$\rightarrow$' according derivation D. The parse tree for D is a hierarchical representation of the pair $\langle \mathbf{x}, \mathbf{y} \rangle$.

In machine translation, the source sentence $\mathbf{x}$ is given and the translation $\mathbf{y}$ is required. A key idea of hierarchical MT is to generate $\mathbf{y}$ by parsing the source sentence $\mathbf{x}$. A derivation D for $\mathbf{x}$ automatically induces a translation $\mathbf{y}$ by applying the rules in D on the target side. Similarly to flat phrase-based MT, Chiang (2005, 2007) used a log-linear model of features to define a score $P(D)$. The highest scoring derivation $\hat{D}$ yields the final translation $\hat{\mathbf{y}}$ ($\langle S, S \rangle \Rightarrow_{\hat{D}} \langle \mathbf{x}, \hat{\mathbf{y}} \rangle$).

Formally, the set $\mathscr{Y}$ of possible translations of $\mathbf{x}$ (i.e. the search space) is defined as:

$$\mathscr{Y} := \{\mathbf{y} | \exists D \in R^+ : \langle S, S \rangle \Rightarrow_D \langle \mathbf{x}, \mathbf{y} \rangle\}. \tag{2.18}$$

An interesting result is that (if the SCFG does not allow unbounded insertions) the language $\mathscr{Y}$ is regular (Allauzen et al., 2014; Iglesias et al., 2011), i.e. we can represent it with a (weighted) finite-state automaton (WFSA). One way to see that is to consider the number of derivations which yield $\mathbf{x}$. Since R is finite and the number of insertions is limited, there is only a finite number of such derivations, so $\mathscr{Y}$ is finite and therefore regular.

2.7.2 FST-based Hierarchical Translation

A popular parsing algorithm for context-free grammars is the CKY algorithm. Chiang (2005, 2007) used CKY parsing with a technique called *cube pruning* to generate translations in hierarchical MT. However, the fact that the search space $\mathscr{Y}$ is a regular language hints towards an implementation using operations on (weighted) finite-state transducers (WFST) such as introduced in Sec. 2.6. HiFST (de Gispert et al., 2010; Iglesias et al., 2009) is a WFST-based decoder for hierarchical phrase-based MT which we will use in some of our experiments in this thesis. HiFST relies heavily on FST operations such as introduced in the previous section and shortest path search to define the decoding process. Rather than keeping the k-best entries in CKY cells like in cube pruning, HiFST constructs lattices in each cell. The lattices are then combined to a recursive transition network (Woods, 1970, RTN) for the whole sentence. An RTN is similar to a WFST but allows non-terminal labels on arcs which reference to sub networks. In a next step, the RTNs are expanded to a push-down automaton or a WFST and rescored with the language model (Allauzen et al., 2014, Fig. 2). The best translation $\hat{\mathbf{y}}$ is found with a shortest path search in the resulting graph.

2.8 Conclusion

We have introduced statistical machine translation (SMT) as a symbolic approach to machine translation: Full source words or phrases are treated as distinct symbols which are translated and reordered on the symbol-level to form a fluent target sentence. This translation and reordering process can be formulated using finite state transducers and formal grammars. FSTs and formal grammars are not only useful for SMT but will also provide the foundation of our hybrid and hierarchical approaches in later chapters.

Sprache, die fur dich dichtet und denkt.
(Language which speaks and thinks for you.)

Victor Klemperer

3 Neural Machine Translation

This chapter is a literature review on neural machine translation. As such, it contains occasional verbatim quotes from the related work sections written by me for the publications in Appendix A that list me as the first author.

3.1 Motivation

In recent years, various fields in the area of natural language processing have been boosted by the rediscovery of neural networks (Goldberg, 2016). However, for a long time, the integration of neural nets into machine translation systems was rather shallow. Early attempts used feedforward neural language models (Bengio et al., 2003, 2006) for the target language to rerank translation lattices (Schwenk et al., 2006). The first neural models which also took the source language into account extended this idea by using the same model with bilingual tuples instead of target language words (Zamora-Martinez et al., 2010), scoring phrase pairs directly with a feedforward net (Schwenk, 2012), or adding a source context window to the neural language model (Devlin et al., 2014; Le et al., 2012). Kalchbrenner and Blunsom (2013) and Cho et al. (2014b) introduced recurrent networks for translation modelling. All those approaches applied neural networks as component in a traditional MT system. Therefore,

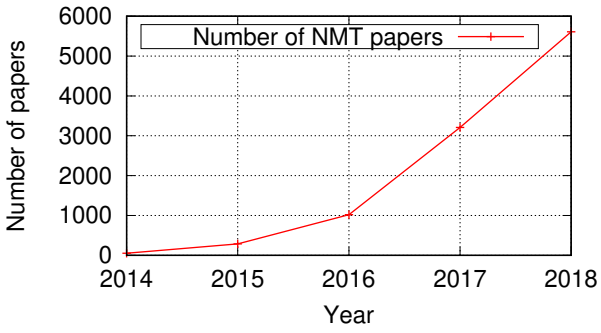


Fig. 3.1 Number of papers mentioning "neural machine translation" per year according Google Scholar.

they retained the log-linear model combination and only exchanged parts in the traditional architecture.

Neural machine translation (NMT) has overcome this separation by using a single large neural net that directly transforms the source sentence into the target sentence (Bahdanau et al., 2015; Cho et al., 2014a; Sutskever et al., 2014). The advent of NMT certainly marks one of the major milestones in the history of MT, and has led to a radical and sudden departure of mainstream research from many previous research lines. This is perhaps best reflected by the explosion of scientific publications related to NMT in the past years (Fig. 3.1).[1] NMT has already been widely adopted in the industry (Crego et al., 2016; Levin et al., 2017; Schmidt and Marg, 2018; Wu et al., 2016b) and is deployed in production systems by Google, Microsoft, Facebook, Amazon, SDL, Yandex, and many more. This chapter will introduce the basic concepts of NMT, and will give a comprehensive overview of current research in the field. Some of the later sections are not strictly required for our own contributions in Ch. 4-8 but are included for the sake of completeness. For even more insight into the field of neural machine translation, we refer the reader to one of the overview papers such as (Cromieres et al., 2017; Koehn, 2017; Neubig, 2017; Popescu-Belis, 2019).

3.2 Word Embeddings

Representing words or phrases as continuous vectors is arguably one of the keys in connectionist models for NLP. To the best of our knowledge, continuous space word representations were first successfully used for language modelling (Bellegarda, 1997; Bengio et al., 2003). The key idea is to represent a word $x \in \Sigma$ as a d-dimensional vector of real numbers. The size d of the

[1] Example Google Scholar search: https://scholar.google.com/scholar?q=%22neural+machine+translation%22&as_ylo=2017&as_yhi=2017

embedding layer is normally chosen to be much smaller than the vocabulary size ($d \ll |\Sigma|$) in order to obtain interesting representations. The mapping from the word to its distributed representation can be represented by an embedding matrix $E \in \mathbb{R}^{d \times |\Sigma|}$ (Collobert and Weston, 2008). The x^{th} column of E (denoted as E_x) holds the d-dimensional representation for the word x.

Learned continuous word representations have the potential of capturing morphological, syntactic and semantic similarity across words (Collobert and Weston, 2008). In neural machine translation, embedding matrices are usually trained jointly with the rest of the network using backpropagation (Rumelhart et al., 1988) and a gradient based optimizer such as stochastic gradient descent. In other areas of NLP, pre-trained word embeddings trained on unlabelled text have become ubiquitous (Collobert et al., 2011). Methods for training word embeddings on raw text often take the context into account in which the word occurs frequently (Mikolov et al., 2013a; Pennington et al., 2014), or use cross-lingual information to improve embeddings (Mikolov et al., 2013b; Upadhyay et al., 2016).

A newly emerging type of *contextualized* word embeddings (McCann et al., 2017; Peters et al., 2017) is gaining popularity in various fields of NLP. Contextualized representations do not only depend on the word itself but on the entire input sentence. Thus, they cannot be described by a single embedding matrix but are usually generated by neural sequence models which have been trained under a language model objective. Most approaches either use LSTM (Peters et al., 2017, 2018) or Transformer architectures (Devlin et al., 2019; Radford et al., 2018) but differ in the way these architectures are used to compute the word representations. Contextualized word embeddings have advanced the state-of-the-art in several NLP benchmarks (Bowman et al., 2018; Devlin et al., 2019; Peters et al., 2018). Goldberg (2019) showed that contextualized embeddings are remarkably sensitive to syntax. Choi et al. (2017b) reported gains from contextualizing word embeddings in NMT using a bag of words.

3.3 Phrase Embeddings

In various NLP tasks such as sentiment analysis or machine translation it is desirable to embed whole phrases or sentences instead of single words. For example, a distributed representation of the source sentence $\mathbf{x}$ could be used as conditional for the distribution over the target sentences $P(\mathbf{y}|\mathbf{x})$. Early approaches to phrase embedding were based on recurrent autoencoders (Pollack, 1990; Socher et al., 2011). To represent a phrase $\mathbf{x} \in \Sigma^I$ as d-dimensional vector, Socher et al. (2011) first trained a word embedding matrix $E \in \mathbb{R}^{d \times |\Sigma|}$. Then, they recursively applied an autoencoder network which finds d-dimensional representations for $2d$-dimensional inputs, where the input is the concatenation of two parent representations. The parent representations

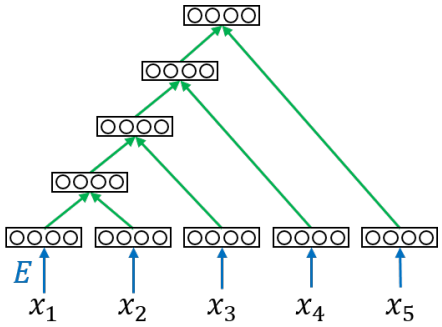


Fig. 3.2 Recursive autoencoder following Socher et al. (2011). The color coding indicates weight sharing.

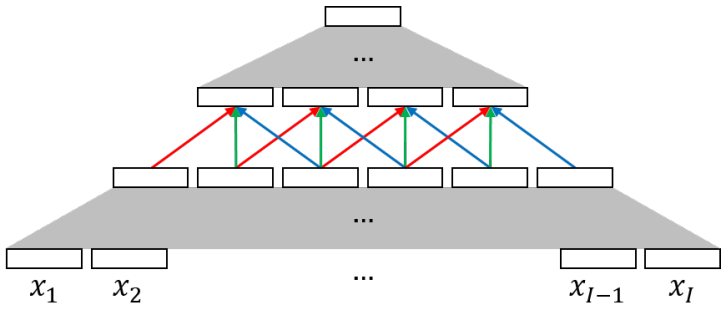


Fig. 3.3 The convolutional sentence model (Kalchbrenner and Blunsom, 2013). The color coding indicates weight sharing.

are either word embeddings or representations calculated by the same autoencoder from two different parents. The order in which representations are merged is determined by a binary tree over $\mathbf{x}$ which can be constructed greedily (Socher et al., 2011) or derived from an Inversion Transduction Grammar (Wu, 1997, ITG) (Li et al., 2013). Fig. 3.2 shows an example of a recurrent autoencoder embedding a phrase with five words into a four dimensional space. One of the disadvantages of recurrent autoencoders is that the word and sentence embeddings need to have the same dimensionality. This restriction is not very critical in sentiment analysis because the distributed sentence representation is only used to extract the sentiment of the writer (Socher et al., 2011). In machine translation, however, the sentence representations need to convey enough information to condition the target sentence distribution on it, and thus should be higher dimensional than the word embeddings.

3.4 Sentence Embeddings

Kalchbrenner and Blunsom (2013) used convolution to find vector representations of phrases or sentences and thus avoided the dimensionality issue of recurrent autoencoders. As shown in

Fig. 3.3, their model yields n-gram representations at each convolution level, with n increasing with depth. The top level can be used as representation for the whole sentence. Other notable examples of using convolution for sentence representations include (dos Santos and Gatti, 2014; Er et al., 2016; Kalchbrenner et al., 2014; Kim, 2014; Mou et al., 2016). However, the convolution operations in these models loose information about the exact word order. and are thus more suitable for sentiment analysis than for tasks like machine translation.[2] A recent line of work uses self-attention rather than convolution to find sentence representations (Shen et al., 2018a; Wu et al., 2018b; Zhang et al., 2018d). Another interesting idea explored by Yu et al. (2018) is to resort to (recursive) relation networks (Palm et al., 2018; Santoro et al., 2017) which repeatedly aggregate pairwise relations between words in the sentence. Recurrent architectures are also commonly used for sentence representation. It has been noted that even random RNNs without any training can work surprisingly well for several NLP tasks (Conneau et al., 2017a, 2018; Wieting and Kiela, 2019).

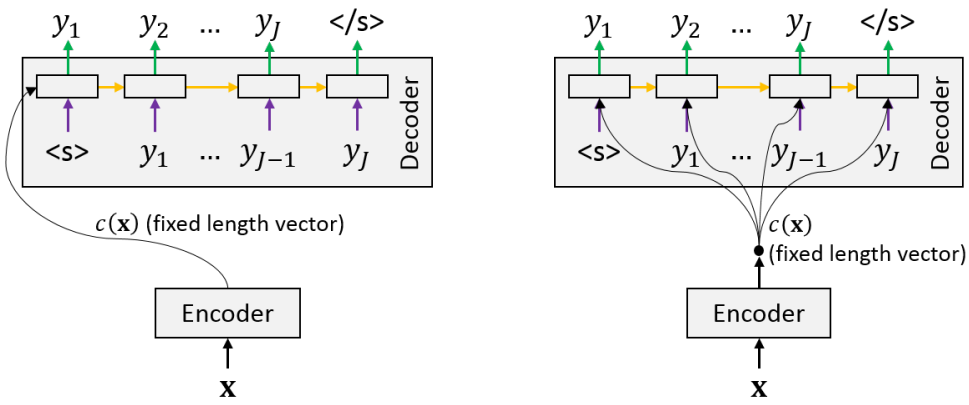
3.5 Encoder-Decoder Networks with Fixed Length Sentence Encodings

Kalchbrenner and Blunsom (2013) were the first who conditioned the target sentence distribution on a distributed fixed-length representation of the source sentence. Their recurrent continuous translation models (RCTM) I and II gave rise to a new family of so-called encoder-decoder networks which is the current prevailing architecture for NMT. Encoder-decoder networks are subdivided into an encoder network which computes a representation of the source sentence, and a decoder network which generates the target sentence from that representation. As introduced in Sec. 1.4 we denote the source sentence as $\mathbf{x} = x_1^I$ and the target sentence as $\mathbf{y} = y_1^J$. All existing NMT models define a probability distribution over the target sentences $P(\mathbf{y}|\mathbf{x})$ by factorizing it into conditionals:

$$P(\mathbf{y}|\mathbf{x}) \overset{\text{Chain rule}}{=} \prod_{j=1}^{J} P(y_j|y_1^{j-1},\mathbf{x}). \tag{3.1}$$

Different encoder-decoder architectures differ vastly in how they model the distribution $P(y_j|y_1^{j-1},\mathbf{x})$. We will first discuss encoder-decoder networks in which the encoder represents the source sentence as a fixed-length vector $c(\mathbf{x})$ like the methods in Sec. 3.4. The conditionals $P(y_j|y_1^{j-1},\mathbf{x})$ are modelled as:

[2] This is not to be confused with convolutional *translation* models which will be reviewed in Sec. 3.6.4



(a) Source sentence encoding is used to initialize the decoder state.

(b) Source sentence encoding is fed to the decoder at each time step.

Fig. 3.4 Encoder-decoder architectures with fixed-length sentence encodings. The color coding indicates weight sharing.

$$P(y_j|y_1^{j-1},\mathbf{x}) = g(y_j|s_j,y_{j-1},c(\mathbf{x})) \tag{3.2}$$

where s_j is the hidden state of a recurrent neural (decoder) network (RNN). We will formally introduce s_j in Sec. 3.6.3. Gated activation functions such as the long short-term memory (Hochreiter and Schmidhuber, 1997, LSTM) or the gated recurrent unit (Cho et al., 2014b, GRU) are commonly used to alleviate the vanishing gradient problem (Hochreiter et al., 2001) which makes it difficult to train RNNs to capture long-range dependencies. Deep architectures with stacked LSTM cells were used by Sutskever et al. (2014). The encoder can be a convolutional network as in the RCTM I (Kalchbrenner and Blunsom, 2013), an LSTM network (Sutskever et al., 2014), or a GRU network (Cho et al., 2014b). $g(\cdot)$ is a feedforward network with a softmax layer at the end which takes as input the decoder state s_j and an embedding of the previous target token y_{j-1}. In addition, $g(\cdot)$ may also take the source sentence encoding $c(\mathbf{x})$ as input to condition on the source sentence (Cho et al., 2014b; Kalchbrenner and Blunsom, 2013). Alternatively, $c(\mathbf{x})$ is just used to initialize the decoder state s_1 (Bahdanau et al., 2015; Sutskever et al., 2014). Fig. 3.4 contrasts both methods. Intuitively, once the source sentence has been encoded, the decoder starts generating the first target sentence symbol y_1 which is then fed back to the decoder network for producing the second symbol y_2. The algorithm terminates when the network produces the end-of-sentence symbol </s>. Sec. 3.7 explains more formally what we mean by the network "generating" a symbol y_j and sheds more light on the aspect of decoding in NMT. Fig. 3.5 shows the complete architecture of Sutskever et al. (2014) who presented one of the first working standalone NMT systems that did not rely on any SMT baseline. One of the reasons why this paper was groundbreaking is the simplicity

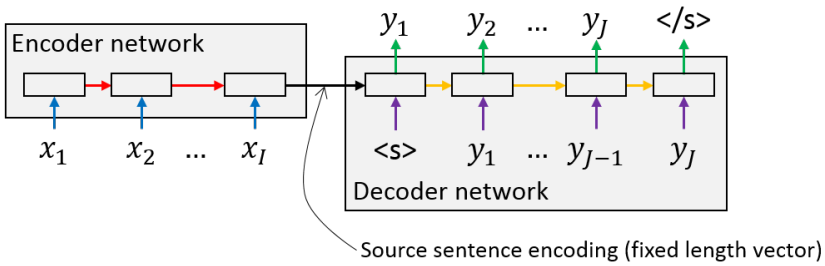
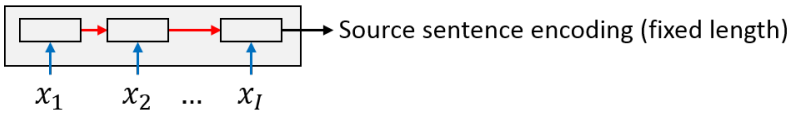
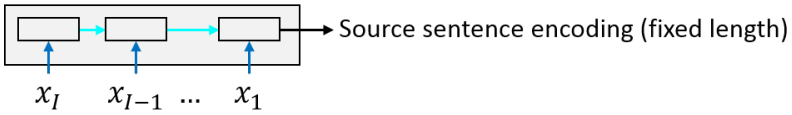


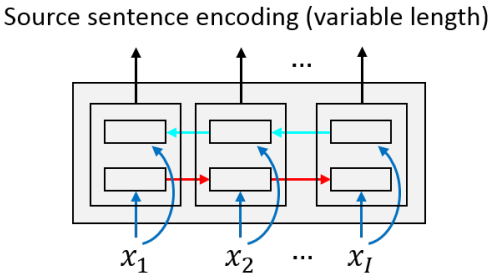
Fig. 3.5 The encoder-decoder architecture of Sutskever et al. (2014). The color coding indicates weight sharing.



(a) Unidirectional encoder used by Cho et al. (2014b).



(b) Reversed unidirectional encoder used by Sutskever et al. (2014).



(c) Bidirectional encoder used by Bahdanau et al. (2015).

Fig. 3.6 Encoder architectures. The color coding indicates weight sharing.

of the architecture, which stands in stark contrast to traditional SMT systems that used a very large number of highly engineered features.

Different ways of providing the source sentence to the encoder network have been explored in the past. Cho et al. (2014b) fed the tokens to the encoder in the natural order they appear in the source sentence (cf. Fig. 3.6a). Sutskever et al. (2014) reported gains from simply feeding the sequence in reversed order (cf. Fig. 3.6b). They argue that these improvements might be "caused by the introduction of many short term dependencies to the dataset" (Sutskever et al., 2014). Bidirectional RNNs (Schuster and Paliwal, 1997, BiRNN) are able to capture both directions (cf. Fig. 3.6c) and are often used in attentional NMT (Bahdanau et al., 2015) as discussed in the next section.

3.6 Attentional Encoder-Decoder Networks

3.6.1 Attention

One problem of early NMT models which is still not fully solved yet (see Sec. 3.10.1) is that they often produced poor translations for long sentences (Sountsov and Sarawagi, 2016). Cho et al. (2014a) suggested that this weakness is due to the fixed-length source sentence encoding. Sentences with varying length convey different amounts of information. Therefore, despite being appropriate for short sentences, a fixed-length vector "does not have enough capacity to encode a long sentence with complicated structure and meaning" (Cho et al., 2014a). Pouget-Abadie et al. (2014) tried to mitigate this problem by chopping the source sentence into short clauses. They composed the target sentence by concatenating the separately translated clauses. However, this approach does not cope well with long-distance reorderings as word reorderings are only possible within a clause. Bahdanau et al. (2015) introduced the concept of *attention* to avoid having a fixed-length source sentence representation. Their model does not use a constant context vector $c(\mathbf{x})$ any more which encodes the whole source sentence. By contrast, the attentional decoder can place its attention only on parts of the source sentence which are useful for producing the next token. The constant context vector $c(\mathbf{x})$ is thus replaced by a series of context vectors $c_j(\mathbf{x})$; one for each time step j.[3]

We will first introduce attention as a general concept before describing the architecture of Bahdanau et al. (2015) in detail in Sec. 3.6.3. We follow the terminology of Vaswani et al. (2017) and describe attention as mapping n query vectors to n output vectors via a mapping table (or a *memory*) of m key-value pairs. This view is related to memory-augmented neural networks which we will discuss in greater detail in Sec. 3.13.3. We make the simplifying assumption that all vectors have the same dimension d so that we can stack the vectors into matrices $Q \in \mathbb{R}^{n\times d}$, $K \in \mathbb{R}^{m\times d}$, and $V \in \mathbb{R}^{m\times d}$. Intuitively, for each query vector we compute an output vector as a weighted sum of the value vectors. The weights are determined by a similarity score between the query vector and the keys (cf. (Vaswani et al., 2017, Eq. 1)):

$$\underbrace{\text{Attention}(K,V,Q)}_{n\times d} = \text{Softmax}(\underbrace{\text{score}(Q,K)}_{n\times m})\underbrace{V}_{m\times d}. \tag{3.3}$$

The output of $\text{score}(Q,K)$ is an $n \times m$ matrix of similarity scores. The softmax function normalizes over the columns of that matrix so that the weights for each query vector sum up to

[3] We refer to j as 'time step' due to the sequential structure of autoregressive models and the left-to-right order of NMT decoding. We note, however, that j does not specify a point in time in the usual sense but rather the position in the target sentence.

Name	Scoring function	Citation
Additive	$\text{score}(Q,K)_{p,q} = v^\intercal \tanh(WQ_{p,:} + UK_{q,:})$	Bahdanau et al. (2015)
Dot-product	$\text{score}(Q,K) = QK^\intercal$	Luong et al. (2015b)
Scaled dot-product	$\text{score}(Q,K) = QK^\intercal d^{-0.5}$	Vaswani et al. (2017)

Table 3.1 Common attention scoring functions. $v \in \mathbb{R}^{d_{\text{att}}}$, $W \in \mathbb{R}^{d_{\text{att}} \times d}$, and $U \in \mathbb{R}^{d_{\text{att}} \times d}$ in additive attention are trainable parameters with d_{att} being the dimensionality of the attention layer.

one. A straight-forward choice for $\text{score}(\cdot)$ proposed by Luong et al. (2015b) is the dot product (i.e. $\text{score}(Q,K) = QK^\intercal$). The most common scoring functions are summarized in Tab. 3.1.

A common way to use attention in NMT is at the interface between encoder and decoder. Bahdanau et al. (2015); Luong et al. (2015b) used the hidden decoder states s_j as query vectors. Both the key and value vectors are derived from the hidden states h_i of a recursive encoder.[4] Formally, this means that $Q = s_j$ are the query vectors , $n = J$ is the target sentence length, $K = V = h_i$ are the key and value vectors, and $m = I$ is the source sentence length.[5] The outputs of the attention layer are used as time-dependent context vectors $c_j(\mathbf{x})$. In other words, rather than using a fixed-length sentence encoding $c(\mathbf{x})$ as in Sec. 3.5, at each time step j we query a memory in which entries store (context-sensitive) representations of the source words. In this setup it is possible to derive an attention matrix $A \in \mathbb{R}^{J \times I}$ to visualize the learned relations between words in the source sentence and words in the target sentence:

$$A := \text{Softmax}(\text{score}((s_j)_{j=1:J}, (h_i)_{i=1:I})). \tag{3.4}$$

Fig. 3.7 shows an example of A from an English-German NMT system with additive attention. The attention matrix captures cross-lingual word relationships such as "is" $\rightarrow$ "ist" or "great" $\rightarrow$ "großer". The system has learned that the English source word "is" is relevant for generating the German target word "ist" and thus emits a high attention weight for this pair. Consequently, the context vector $c_j(\mathbf{x})$ at time step $j = 3$ mainly represents the source word "is" ($c_3(\mathbf{x}) \approx h_2$). This is particularly significant as the system was not explicitly trained to align words but to optimize translation performance. However, as we will argue in Sec. 8.4, it would be wrong to think of A as a soft version of a traditional SMT word alignment like described in Sec. 2.4.

An important generalization of attention is *multi-head* attention proposed by Vaswani et al. (2017). The idea is to perform H attention operations instead of a single one where H is the number of attention heads (usually $H = 8$). The query, key, and value vectors for the

[4] s_j and h_i are defined in Sec. 3.5 and Sec. 3.6.3.

[5] An exception is the model of Mino et al. (2017) that splits h_i into two parts and uses the first part as key and the second as value.

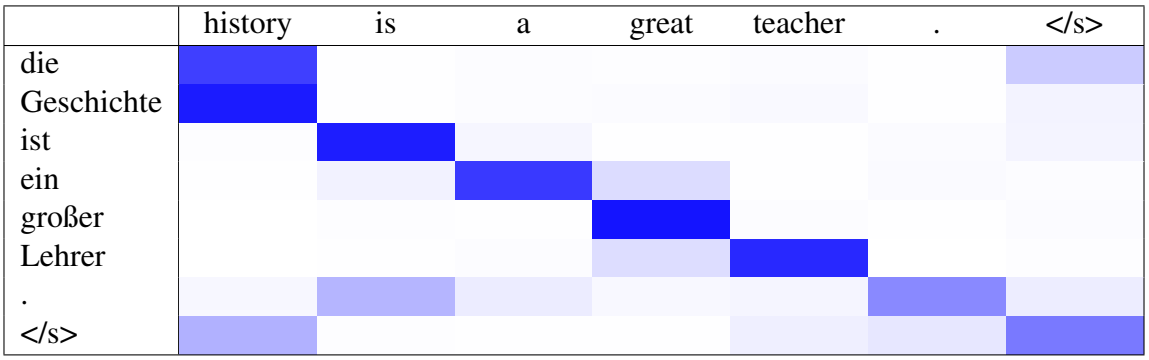


Fig. 3.7 Attention weight matrix A for the translation from the English sentence "history is a great teacher ." to the German sentence "die Geschichte ist ein großer Lehrer .". Dark shades of blue indicate high attention weights.

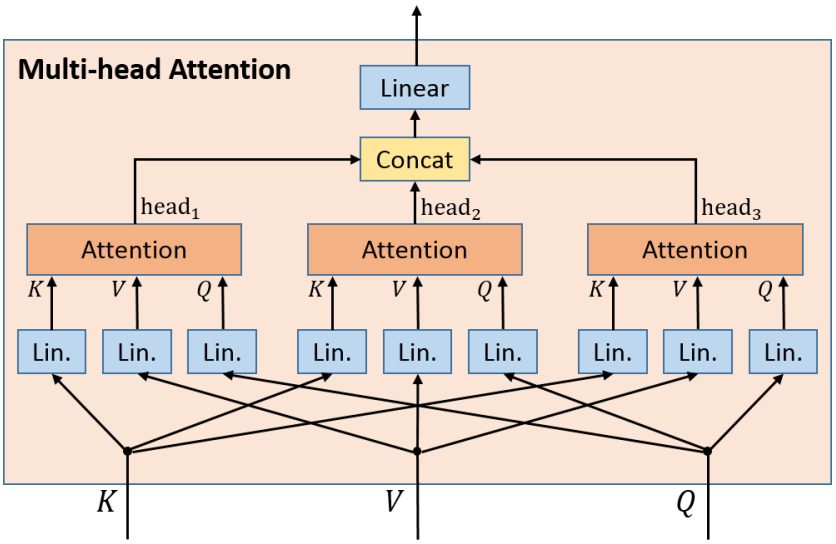


Fig. 3.8 Multi-head attention with three attention heads.

attention heads are linear transforms of Q, K, and V. The output of multi-head attention is the concatenation of the outputs of each attention head. The dimensionality of the attention heads is usually divided by H to avoid increasing the number of parameters. Formally, it can be described as follows (Vaswani et al., 2017):

$$\text{MultiHeadAttention}(K,V,Q) = \text{Concat}(\text{head}_1,\ldots,\text{head}_H)W^O \tag{3.5}$$

with weight matrix $W^O \in \mathbb{R}^{d\times d}$ where

$$\text{head}_h = \text{Attention}(KW_h^K, VW_h^V, QW_h^Q) \tag{3.6}$$

the	first	cold	shower	<pad>	<pad>
even	the	monkey	seems	to	want
a	little	coat	of	straw	<pad>

Fig. 3.9 A tensor containing a batch of three source sentences of different lengths ("the first cold shower", "even the monkey seems to want", "a little coat of straw" – a haiku by Basho (Basho and Reichhold, 2013)). Short sentences are padded with <pad>. The training loss and attention masks are visualized with green (enabled) and red (disabled) background.

with weight matrices $W_h^K, W_h^V, W_h^Q \in \mathbb{R}^{d \times \frac{d}{H}}$ for $h \in [1, H]$. Fig. 3.8 shows a multi-head attention module with three heads. Note that with multi-head attention it is not obvious anymore how to derive a single attention weight matrix A like shown in Fig. 3.7. Therefore, models using multi-head attention tend to be more difficult to interpret.

The concept of attention is no longer just a technique to improve sentence lengths in NMT. Since its introduction by Bahdanau et al. (2015) it has become a vital part of various NMT architectures, culminating in the Transformer architecture (Sec. 3.6.5) which is entirely attention-based. Attention has also been proven effective for, inter alia, object recognition (Ba et al., 2014; Larochelle and Hinton, 2010; Mnih et al., 2014), image caption generation (Xu et al., 2015), video description (Yao et al., 2015), speech recognition (Chan et al., 2016; Chorowski et al., 2014), cross-lingual word-to-phone alignment (Duong et al., 2016), bioinformatics (Sønderby et al., 2015), text summarization (Rush et al., 2015), text normalization (Sproat and Jaitly, 2016), grammatical error correction (Yuan and Briscoe, 2016), question answering (Hermann et al., 2015; Sukhbaatar et al., 2015; Yang et al., 2016b), natural language understanding and inference (Dong and Lapata, 2016; Im and Cho, 2017; Liu et al., 2016c; Shen et al., 2018a), uncertainty detection (Adel and Schütze, 2017), photo optical character recognition (Lee and Osindero, 2016), and natural language conversation (Shang et al., 2015).

3.6.2 Attention Masks and Padding

NMT usually groups sentences into batches to make more efficient use of the available hardware and to reduce noise in gradient estimation (cf. Sec. 3.11.1). However, the central data structure for many machine learning frameworks (Abadi et al., 2016; Bastien et al., 2012) are *tensors* – multi-dimensional arrays with fixed dimensionality. Re-arranging source sentences as tensor often results in some unused space as the sentences may vary in length. In practice, shorter sentences are filled up with a special padding symbol <pad> to match the length of the longest sentence in the batch (Fig. 3.9). Most implementations work with masks to avoid taking padded positions into account when computing the training loss. Attention layers also have to be

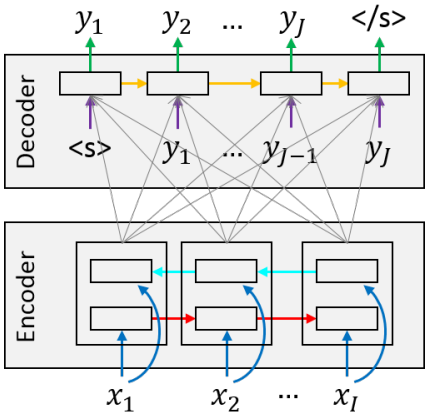


Fig. 3.10 The RNNsearch model following Bahdanau et al. (2015). The color coding indicates weight sharing. Gray arrows represent attention.

restricted to non-padding symbols which is also usually realized by multiplying the attention weights by a mask that sets the attention weights for padding symbols to zero.

3.6.3 Recurrent Neural Machine Translation

This section contains a complete formal description of the RNNsearch architecture of Bahdanau et al. (2015) which was the first NMT model using attention. Recall that NMT uses the chain rule to decompose the probability $P(\mathbf{y}|\mathbf{x})$ of a target sentence $\mathbf{y} = y_1^J$ given a source sentence $\mathbf{x} = x_1^I$ into left-to-right conditionals (Eq. 3.1). RNNsearch models the conditionals as follows (Bahdanau et al., 2015, Eq. 2,4):

$$P(\mathbf{y}|\mathbf{x}) \overset{\text{Eq. 3.1}}{=} \prod_{j=1}^{J} P(y_j|y_1^{j-1},\mathbf{x}) = \prod_{j=1}^{J} g(y_j|y_{j-1},s_j,c_j(\mathbf{x})). \tag{3.7}$$

Similarly to Eq. 3.2, the function $g(\cdot)$ encapsulates the decoder network which computes the distribution for the next target token y_j given the last produced token y_{j-1}, the RNN decoder state $s_j \in \mathbb{R}^n$, and the context vector $c_j(\mathbf{x}) \in \mathbb{R}^m$. The sizes of the encoder and decoder hidden layers are denoted with m and n. The context vector $c_j(\mathbf{x})$ is a distributed representation of the relevant parts of the source sentence. In NMT without attention (Cho et al., 2014b; Sutskever et al., 2014) (Sec. 3.5), the context vector is constant and thus needs to encode the whole source sentence. Adding an attention mechanism results in different context vectors for each target sentence position j. This effectively addresses issues in NMT due to the limited capacity of a fixed context vector as illustrated in Fig. 3.10.

As outlined in Sec. 3.6.1, the context vectors $c_j(\mathbf{x})$ are weighted sums of source sentence *annotations* $\mathbf{h} = (h_1, \ldots, h_I)$. The annotations are produced by the encoder network. In other words, the encoder converts the input sequence $\mathbf{x}$ to a sequence of annotations $\mathbf{h}$ of the same length. Each annotation $h_i \in \mathbb{R}^m$ encodes information about the entire source sentence $\mathbf{x}$ "with a strong focus on the parts surrounding the i-th word of the input sequence" (Bahdanau et al., 2015, Sec. 3.1). RNNsearch uses a bidirectional RNN (Schuster and Paliwal, 1997, BiRNN) to generate the annotations. A BiRNN consists of two independent RNNs. The forward RNN $\overrightarrow{f}$ reads $\mathbf{x}$ in the original order (from x_1 to x_I). The backward RNN $\overleftarrow{f}$ consumes $\mathbf{x}$ in reversed order (from x_I to x_1):

$$\overrightarrow{h}_i = \overrightarrow{f}(x_i, \overrightarrow{h}_{i-1}) \tag{3.8}$$

$$\overleftarrow{h}_i = \overleftarrow{f}(x_i, \overleftarrow{h}_{i+1}). \tag{3.9}$$

The RNNs $\overrightarrow{f}(\cdot)$ and $\overleftarrow{f}(\cdot)$ are usually LSTM (Hochreiter and Schmidhuber, 1997) or GRU (Cho et al., 2014b) cells. The annotation h_i is the concatenation of the hidden states $\overrightarrow{h}_i$ and $\overleftarrow{h}_i$ (Bahdanau et al., 2015, Sec. 3.2):

$$h_i = [\overrightarrow{h}_i^\intercal ; \overleftarrow{h}_i^\intercal]^\intercal. \tag{3.10}$$

The context vectors $c_j(\mathbf{x}) \in \mathbb{R}^m$ are computed from the annotations as weighted sum with weights $\alpha_j \in [0,1]^I$ (Bahdanau et al., 2015, Eq. 5):

$$c_j(\mathbf{x}) = \sum_{i=1}^{I} \alpha_{j,i} h_i. \tag{3.11}$$

The weights are determined by the alignment model $a(\cdot)$:

$$\alpha_{j,i} = \frac{1}{Z} \exp(a(s_{j-1}, h_i)) \text{ with } Z = \sum_{k=1}^{I} \exp(a(s_{j-1}, h_k)) \tag{3.12}$$

where $a(s_{j-1}, h_i)$ is a feedforward neural network which estimates the importance of annotation h_i for producing the j-th target token given the current decoder state $s_{j-1} \in \mathbb{R}^n$. In the terminology of Sec. 3.6.1, h_i represent the keys and values, s_j are the queries, and $a(\cdot)$ is the attention scoring function.

The function $g(\cdot)$ in Eq. 3.7 does not only take the previous target token y_{j-1} and the context vector c_j but also the decoder hidden state s_j.

$$s_j = f(s_{j-1}, y_{j-1}, c_j) \tag{3.13}$$

where $f(\cdot)$ is modelled by a GRU or LSTM cell. The function $g(\cdot)$ is defined as follows.

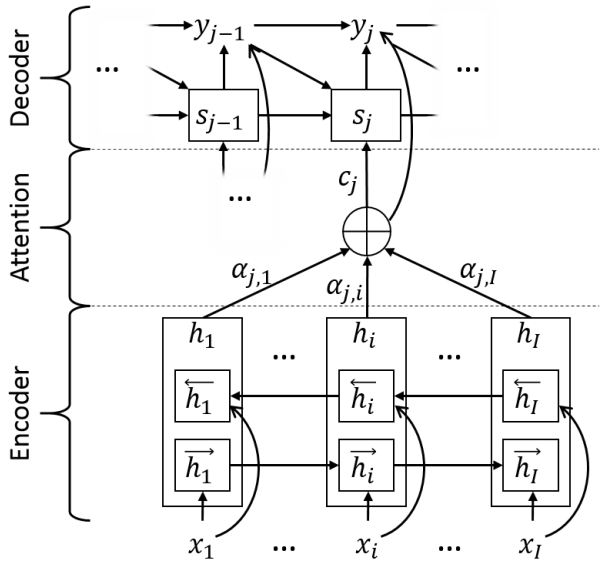


Fig. 3.11 Illustration of the attention mechanism in RNNsearch (Bahdanau et al., 2015).

$$g(y_j|y_{j-1},s_j,c_j) \propto \exp(W_o \max(t_j,u_j)) \tag{3.14}$$

with

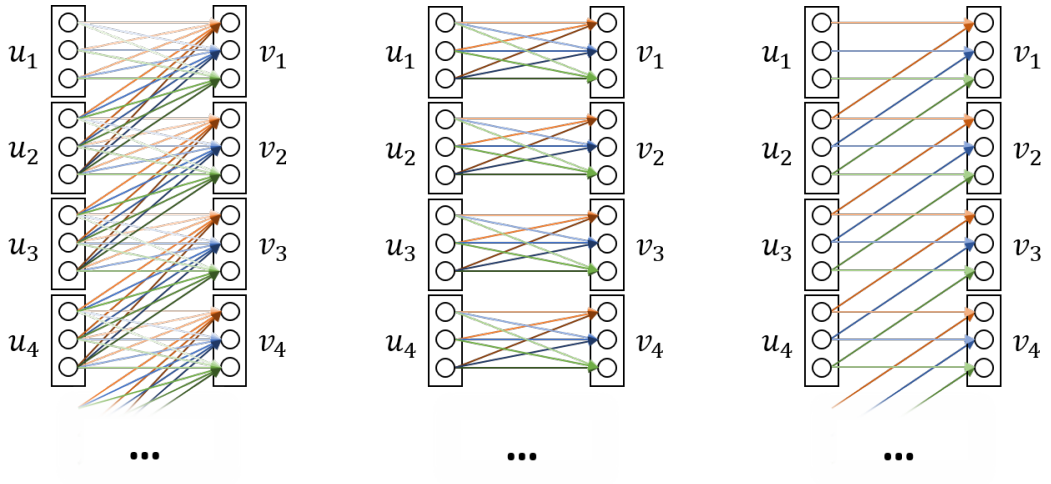
$$t_j = T_s s_j + T_y E y_{j-1} + T_c c_j \tag{3.15}$$

$$u_j = U_s s_j + U_y E y_{j-1} + U_c c_j \tag{3.16}$$

where $\max(\cdot)$ is the *element-wise* maximum, and $W_o \in \mathbb{R}^{|\Sigma_{trg}| \times l}$, $T_s, U_s \in \mathbb{R}^{l \times n}$, $T_y, U_y \in \mathbb{R}^{l \times k}$, $E \in \mathbb{R}^{k \times |\Sigma_{trg}|}$, $T_c, U_c \in \mathbb{R}^{l \times m}$ are weight matrices. The definition of $g(\cdot)$ can be seen as connecting the output of the recurrent layer, an k-dimensional embedding of the previous target token, and the context vector with a single maxout layer (Goodfellow et al., 2013b) of size l and using a softmax over the target language vocabulary (Bahdanau et al., 2015). Fig. 3.11 illustrates the complete RNNsearch model.

3.6.4 Convolutional Neural Machine Translation

Although convolutional neural networks (CNNs) have first been proposed by Waibel et al. (1989) for phoneme recognition, their traditional use case is computer vision (LeCun et al., 1989a, 1990, 1998). CNNs are especially useful for processing images because of two reasons. First, they use a high degree of weight tying and thus reduce the number of parameters dramatically compared to fully connected networks. This is crucial for high dimensional input like visual imagery. Second, they automatically learn space invariant features. Spatial



(a) Standard convolution. (b) Pointwise convolution. (c) Depthwise convolution.

Fig. 3.12 Types of 1D-convolution used in NMT. The color coding indicates weight sharing.

invariance is desirable in vision since we often aim to recognize objects or features regardless of their exact position in the image. In NLP, convolutions are usually one dimensional since we are dealing with sequences rather than two dimensional images as in computer vision. We will therefore limit our discussions to the one dimensional case. We will also exclude concepts like pooling or strides as they are uncommon for sequence models in NLP.

The input to an 1D convolutional layer is a sequence of M-dimensional vectors $u_1, \dots, u_I$. The literature about CNNs usually refers to the M dimensions in each $u_i \in \mathbb{R}^M$ ($i \in [1, I]$) as *channels*, and to the i-axis as *spatial dimension*. The convolution transforms the input sequence $u_1, \dots, u_I$ to an output sequence of N-dimensional $v_1, \dots, v_I$ of the same length by moving a *kernel* of width K over the input sequence. The kernel is a linear transform which maps the K-gram $u_i, \dots, u_{i+K-1}$ to the output v_i for $i \in [1, I]$ (we append $K-1$ padding symbols to the input). Standard convolution parameterizes this linear transform with a full weight matrix $W^{\text{std}} \in \mathbb{R}^{KM \times N}$:

$$\text{StdConv:}(v_i)_n = \sum_{m=1}^{M} \sum_{k=0}^{K-1} W^{\text{std}}_{kM+m,n} (u_{i+k})_m \tag{3.17}$$

with $i \in [1, I]$ and $n \in [1, N]$. Standard convolution represents two kinds of dependencies: Spatial dependency (inner sum in Eq. 3.17) and cross-channel dependency (outer sum in Eq. 3.17). Pointwise and depthwise convolution factor out these dependencies into two separate operations:

$$\text{PointwiseConv:}(v_i)_n = \sum_{m=1}^{M} W^{\text{pw}}_{m,n} (u_i)_m = u_i W^{\text{pw}} \tag{3.18}$$

Name	Number of parameters
Standard convolution	KMN
Pointwise convolution	MN
Depthwise convolution	KN
Depthwise separable convolution	N(M+K)

Table 3.2 Types of convolution and their number of parameters.

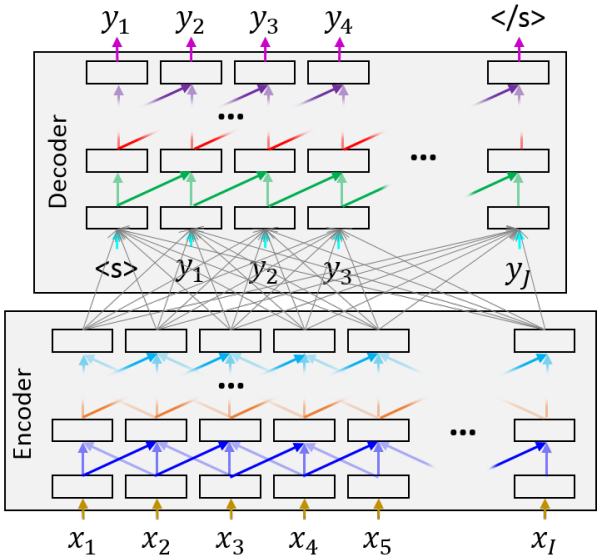


Fig. 3.13 NMT with a convolutional encoder and a convolutional decoder like in the ConvS2S architecture (Gehring et al., 2017b). The color coding indicates weight sharing. Gray arrows represent attention.

$$\text{DepthwiseConv:}(v_i)_n = \sum_{k=0}^{K-1} W_{k,n}^{\text{dw}} (u_{i+k})_n \tag{3.19}$$

where $W^{\text{pw}} \in \mathbb{R}^{M \times N}$ and $W^{\text{dw}} \in \mathbb{R}^{K \times N}$ are weight matrices. Fig. 3.12 illustrates the differences between these types of convolution. The idea behind *depthwise separable* convolution is to replace standard convolutional with depthwise convolution followed by pointwise convolution. As shown in Tab. 3.2, the decomposition into two simpler steps reduces the number of parameters and has been shown to make more efficient use of the parameters than regular convolution in vision (Chollet, 2017; Howard et al., 2017).

Using convolution rather than recurrence in NMT models has several potential advantages. First, they reduce sequential computation and are therefore easier parallelizable on GPU hardware. Second, their hierarchical structure connects distant words via a shorter path than sequential topologies (Gehring et al., 2017b) which eases learning (Hochreiter et al.,

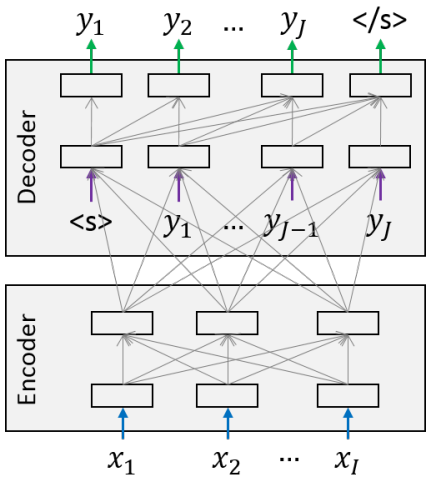


Fig. 3.14 Purely attention-based NMT as proposed by Vaswani et al. (2017) with two layers. The color coding indicates weight sharing. Gray arrows represent attention.

2001). Both regular (Gehring et al., 2017a,b; Kalchbrenner et al., 2016) and depthwise separable (Kaiser et al., 2017; Wu et al., 2019a) convolution have been used for NMT in the past. Fig. 3.13 shows the general architecture for a fully convolutional NMT model such as ConvS2S (Gehring et al., 2017b) or SliceNet (Kaiser et al., 2017) in which both encoder and decoder are convolutional. Stacking multiple convolutional layers increases the effective context size. In the decoder, we need to mask the receptive field of the convolution operations to make sure that the network has no access to future information (van den Oord et al., 2016b). Encoder and decoder are connected via attention. Gehring et al. (2017b) used attention into the encoder representations after each convolutional layer in the decoder.

3.6.5 Self-attention-based Neural Machine Translation

Recall that Eq. 3.1 states that NMT factorizes $P(\mathbf{y}|\mathbf{x})$ into conditionals $P(y_j|y_1^{j-1},\mathbf{x})$. We have reviewed two ways to model the dependency on the source sentence $\mathbf{x}$ in NMT: via a fixed-length sentence encoding $c(\mathbf{x})$ (Sec. 3.5) or via time-dependent context vectors $c_j(\mathbf{x})$ which are computed using attention (Sec. 3.6.1). We have also presented two ways to implement the dependency on the target sentence prefix y_1^{j-1}: via a recurrent connection which passes through the decoder state to the next time step (Sec. 3.6.3) or via convolution (Sec. 3.6.4). A third option to model target side dependency is using *self-attention*. Using the terminology introduced in Sec. 3.6.1, decoder self-attention derives all three components (queries, keys, and values) from the decoder state. The decoder conditions on the translation prefix y_1^{j-1} by attending to its own states from previous time steps. Besides machine translation, self-attention has been applied to various NLP tasks such as sentiment analysis (Cheng et al., 2016a), natural language

inference (Liu et al., 2016c; Parikh et al., 2016; Shen et al., 2018a,b), text summarization (Paulus et al., 2017), headline generation (Daniil et al., 2019), sentence embedding (Lin et al., 2017; Wu et al., 2018b; Zhang et al., 2018d), and reading comprehension (Hu et al., 2018). Similarly to convolution, self-attention introduces short paths between distant words and reduces the amount of sequential computation. Studies indicate that these short paths are especially useful for learning strong semantic feature extractors, but (perhaps somewhat counter-intuitively) less so for modelling long-range subject-verb agreement (Tang et al., 2018a). Like in convolutional models we also need to mask future decoder states to prevent conditioning on future tokens (cf. Sec. 3.6.2). The general layout for self-attention-based NMT models is shown in Fig. 3.14. The first example of this new class of NMT models was the Transformer (Vaswani et al., 2017). The Transformer uses attention for three purposes: 1) within the encoder to enable context-sensitive word representations which depend on the whole source sentence, 2) between the encoder and the decoder as in previous models, and 3) within the decoder to condition on the current translation history. The Transformer uses multi-head attention (Sec. 3.6.1) rather than regular attention. Using multi-head attention has been shown to be essential for the Transformer architecture (Chen et al., 2018b; Tang et al., 2018a).

A challenge in self-attention-based models (and to some extent in convolutional models) is that vanilla attention as introduced in Sec. 3.6.1 by itself has no notion of order. The key-value pairs in the memory are accessed purely based on the correspondence between key and query (*content-based* addressing) and not based on a location of the key in the memory (*location-based*).[6] This is less of a problem in recurrent NMT (Sec. 3.6.3) as queries, keys, and values are derived from RNN states and already carry a strong sequential signal due to the RNN topology. In the Transformer architecture, however, recurrent connections are removed in favor of attention. Vaswani et al. (2017) tackled this problem using *positional encodings*. Positional encodings are (potentially partial) functions $\mathrm{PE} : \mathbb{N} \nrightarrow \mathbb{R}^D$ where D is the word embedding size, i.e. they are D-dimensional representations of natural numbers. They are added to the (input and output) word embeddings to make them (and consequently the queries, keys, and values) position-sensitive. Vaswani et al. (2017) stacked sine and cosine functions of different frequencies to implement $\mathrm{PE}(\cdot)$:

$$\mathrm{PE}_{\sin}(n)_d = \begin{cases} \sin(10000^{-\frac{d}{D}} n) & : d \text{ is even} \\ \cos(10000^{-\frac{d}{D}} n) & : d \text{ is odd} \end{cases} \tag{3.20}$$

[6] We will discuss cases in which both content and location are taken into account in Secs. 3.13.2 and 3.13.3

for $n \in \mathbb{N}$ and $d \in [1, D]$. Alternatively, positional encodings can be learned in an embedding matrix (Gehring et al., 2017b):

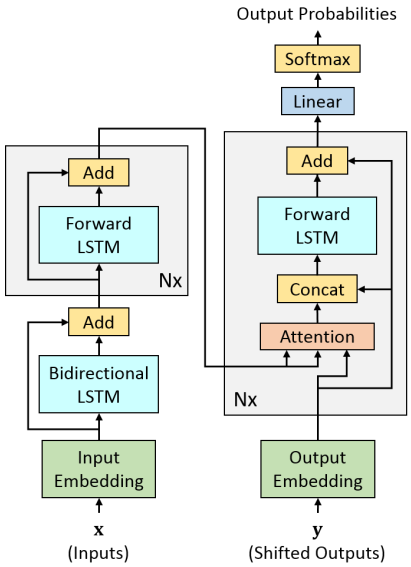
$$\text{PE}_{\text{learned}}(n) = W_{:,n} \tag{3.21}$$

with weight matrix $W \in \mathbb{R}^{d \times N}$ for some sufficiently large N. The input to $\text{PE}(\cdot)$ is usually the absolute position of the word in the sentence (Gehring et al., 2017b; Vaswani et al., 2017), but relative positioning is also possible (Shaw et al., 2018). We will give an overview of extensions to the Transformer architecture in Sec. 3.13.1.

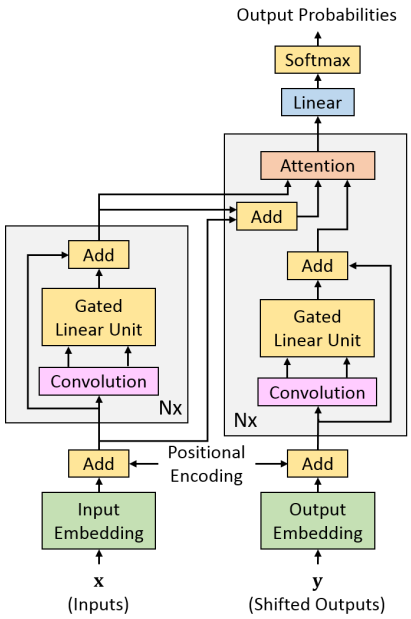
3.6.6 Comparison of the Fundamental Architectures

As outlined in the previous sections, NMT can come in one of three flavors: recurrent, convolutional, or self-attention-based. In this section, we will discuss three concrete architectures in greater detail – one of each flavor. Our own empirical results that compare and combine these architectures are described in Sec 4.6.

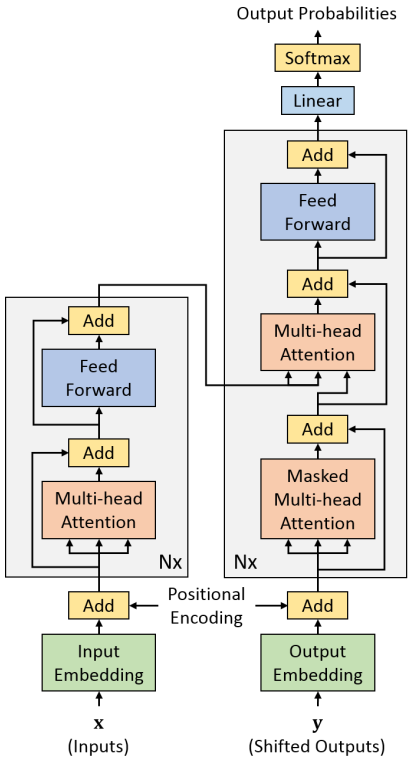
Fig. 3.15 visualizes the data streams in Google's Neural Machine Translation system (Wu et al., 2016b, GNMT) as example of a recurrent network, the convolutional ConvS2S model (Gehring et al., 2017b), and the self-attention-based Transformer model (Vaswani et al., 2017) in plate notation. We excluded components like dropout (Srivastava et al., 2014), batch normalization (Ioffe and Szegedy, 2015), and layer normalization (Ba et al., 2016) to simplify the diagrams. All models fall in the general category of encoder-decoder networks, with the encoder in the left column and the decoder in the right column. Output probabilities are generated by a linear projection layer followed by a softmax activation at the end. They all use attention at each decoder layer to connect the encoder with the decoder, although the specifics differ. GNMT (Fig. 3.15a) uses regular attention, ConvS2S (Fig. 3.15b) adds the source word encodings to the values, and the Transformer (Fig. 3.15c) uses multi-head attention (Sec. 3.6.1). Residual connections (He et al., 2016c) are used in all three architectures to encourage gradient flow in multi-layer networks. Positional encodings are used in ConvS2S and the Transformer, but not in GNMT. An interesting fusion is the RNMT+ model (Chen et al., 2018b) shown in Fig. 3.15d which reintroduces ideas from the Transformer like multi-head attention into recurrent NMT. Other notable mixed architectures include Gehring et al. (2017a) who used a convolutional encoder with a recurrent decoder, Miculicich et al. (2018a); Wang et al. (2019a); Werlen et al. (2018) who added self-attention connections to a recurrent decoder, Hao et al. (2019) who used a Transformer encoder and a recurrent encoder in parallel, and Lin et al. (2018b) who equipped a recurrent decoder with a convolutional decoder to provide global target-side context.



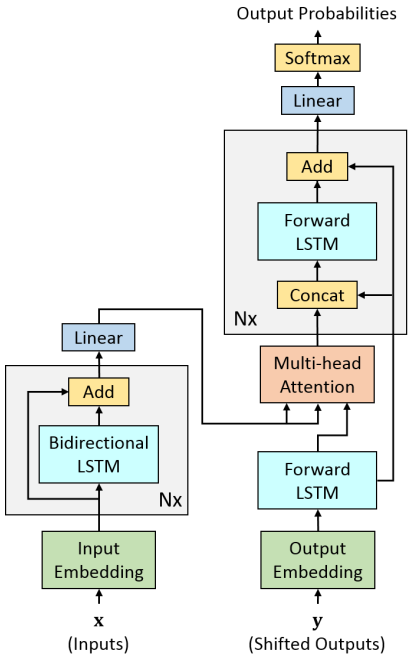
(a) GNMT (Wu et al., 2016b).



(b) ConvS2S (Gehring et al., 2017b).



(c) Transformer (Vaswani et al., 2017).



(d) RNMT+ (Chen et al., 2018b).

Fig. 3.15 Comparison of NMT architectures. The three inputs to attention modules are (from left to right): keys (K), values (V), and queries (Q) as in Fig. 3.8.

3.7 Neural Machine Translation Decoding

3.7.1 The Search Problem in NMT

So far we have described how NMT defines the translation probability $P(\mathbf{y}|\mathbf{x})$. However, in order to apply these definitions directly, both the source sentence $\mathbf{x}$ and the target sentence $\mathbf{y}$ have to be given. They do not directly provide a method for generating a target sentence $\mathbf{y}$ from a given source sentence $\mathbf{x}$ which is the ultimate goal in machine translation. The task of finding the most likely translation $\hat{\mathbf{y}}$ for a given source sentence $\mathbf{x}$ is known as the *decoding* or *inference* problem:

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y} \in \Sigma_{trg}^*} P(\mathbf{y}|\mathbf{x}). \tag{3.22}$$

NMT decoding is non-trivial for mainly two reasons. First, the search space is vast as it grows exponentially with the sequence length. For example, if we assume a common vocabulary size of $|\Sigma_{trg}| = 32,000$, there are already more possible translations with 20 words or less than atoms in the observable universe ($32,000^{20} \gg 10^{82}$). Thus, complete enumeration of the search space is impossible. Second, as we will see in Sec. 3.10, certain types of model errors are very common in NMT. The mismatch between the most likely and the "best" translation has deep implications on search as more exhaustive search often leads to worse translations. We will discuss possible solutions to both problems in the remainder of Sec. 3.7. Sec. 5.4.4 presents our own investigations into the problem of NMT search errors and model errors.

3.7.2 Greedy and Beam Search

The most popular decoding algorithms for NMT are greedy search and beam search. Both search procedures are based on the left-to-right factorization of NMT in Eq. 3.1. Translations are built up from left to right while partial translation prefixes are scored using the conditionals $P(y_j|y_1^{j-1}, \mathbf{x})$. This means that both algorithms work in a time-synchronous manner: in each iteration j, partial hypotheses of (up to) length j are compared to each other, and a subset of them is selected for expansion in the next time step. The algorithms terminate if either all or the best of the selected hypotheses end with the end-of-sentence symbol </s> or if some maximum number of iterations is reached. Fig. 3.16 illustrates the difference between greedy search and beam search. Greedy search (highlighted in green) selects the single best expansion at each time step: 'c' at $j = 1$, 'a' at $j = 2$, and 'b' at $j = 3$. However, greedy search is vulnerable to the so-called *garden-path problem* (Koehn, 2017). The algorithm selects 'c' in the first time step which turns out to be a mistake later on as subsequent distributions are very smooth and scores are comparably low. However, greedy decoding cannot correct this mistake later as

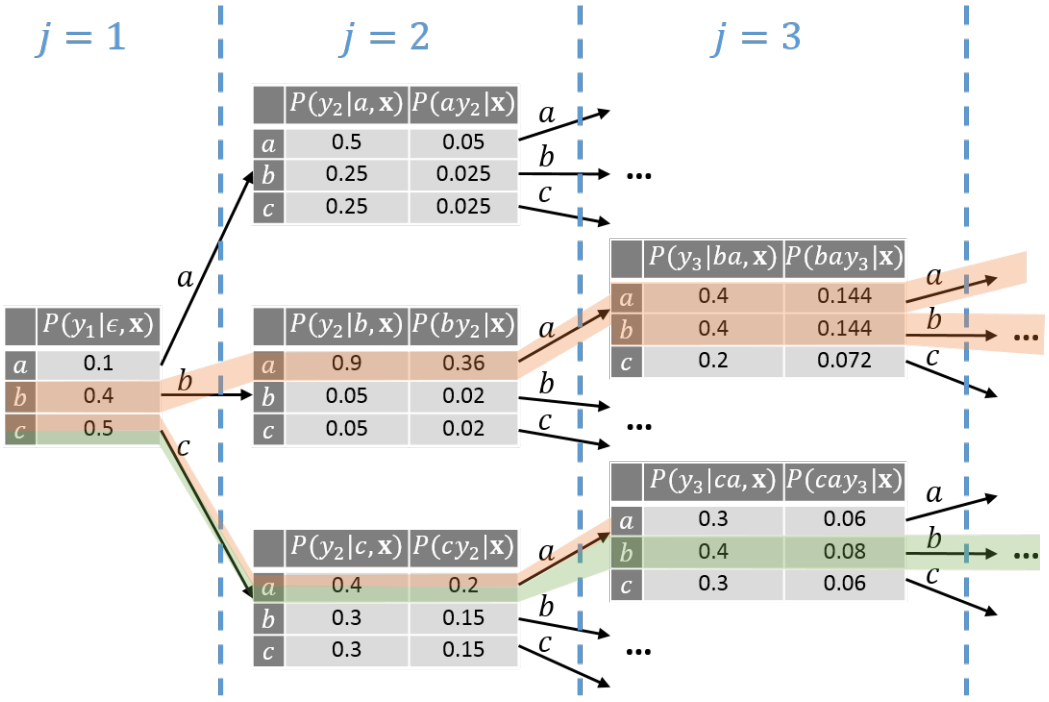


Fig. 3.16 Comparison between greedy (highlighted in green) and beam search (highlighted in orange) with beam size 2.

it is already committed to this path. Beam search (highlighted in orange in Fig. 3.16) tries to mitigate the risk of the garden-path problem by passing not one but n possible translation prefixes to the next time step ($n = 2$ in Fig. 3.16). The n hypotheses which survive a time step are called *active hypotheses*. At each time step, the accumulated path scores for all possible continuations of active hypotheses are compared, and the n best ones are selected. Thus, beam search does not only expand 'c' but also 'b' in time step 1, and thereby finds the high scoring translation prefix 'ba'. Note that although beam search seems to be the more accurate search procedure, it is not guaranteed to always find a translation with higher or equal score as greedy decoding.[7] It is therefore still prone to the garden-path problem, although less so than greedy search.

3.7.3 Formal Description of Decoding for the RNNsearch Model

In this thesis, we generally view decoding in NMT as (approximated) shortest path search problem on a weighted graph structure as explained in Sec. 5.4.1. This view will enable us to use a wide range of additional search algorithms, and provide a strong formal framework

[7] For example, imagine a series of high entropy conditionals after 'baa' and low entropy conditionals after 'cab' in Fig. 3.16

Algorithm 1 OneStepRNNsearch$(s_{prev}, y_{prev}, \mathbf{h})$

1: $\alpha \overset{Eq.\ 3.12}{\leftarrow} \frac{1}{Z}[\exp(a(s_{prev}, h_i))]_{i\in[1,I]}$ {Attention weights ($\alpha \in \mathbb{R}^I$, Z as in Eq. 3.12)}
2: $c \overset{Eq.\ 3.11}{\leftarrow} \sum_{i=1}^{I} \alpha_i \cdot h_i$ {Context vector update ($c \in \mathbb{R}^m$)}
3: $s \overset{Eq.\ 3.13}{\leftarrow} f(s_{prev}, y_{prev}, c)$ {RNN state update ($s \in \mathbb{R}^n$)}
4: $p \overset{Eq.\ 3.1}{\leftarrow} g(y_{prev}, s, c)$ {$p \in \mathbb{R}^{|\Sigma_{trg}|}$ is the distribution over the next target token $P(y_j|\cdot)$}
5: **return** s, p

Algorithm 2 GreedyRNNsearch$(s_{init}, \mathbf{h})$

1: $\mathbf{y} \leftarrow \langle\rangle$
2: $s \leftarrow s_{init}$
3: $y \leftarrow$ <s>
4: **while** $y \neq$ </s> **do**
5: $\quad s, p \leftarrow$ OneStepRNNsearch$(s, y, \mathbf{h})$
6: $\quad y \leftarrow \arg\max_{w \in \Sigma_{trg}} \pi_w(p)$
7: $\quad \mathbf{y}$.append(y)
8: **end while**
9: **return** $\mathbf{y}$

for combining NMT with different kinds of other models. However, NMT literature usually describes search on a lower abstraction level which relates more directly to NMT (Koehn, 2017; Neubig, 2017). In this section, we will formally introduce decoding for the RNNsearch model (Bahdanau et al., 2015) in this way. We will resort to the mathematical symbols used in Sec. 3.6.3 to describe the algorithms.

In pure NMT systems, a simple left-to-right beam search with a small beam is normally used for decoding. First, the source annotations $\mathbf{h}$ are computed and stored as this does not require any search. Then, we compute the distribution for the first target token y_1 using OneStepRNNsearch$(s_{init}, <s>, \mathbf{h})$ (Alg. 1). The initial decoder state s_{init} is often a linear transform of the last encoder hidden state h_I: $s_{init} = W h_I$ for some weight matrix $W \in \mathbb{R}^{n\times m}$.

Greedy decoding selects the most likely target token according the returned distribution and iteratively calls OneStepRNNsearch$(\cdot)$ until the end-of-sentence symbol </s> is emitted (Alg. 2). We use the projection function $\pi_w(p)$ (Eq. 1.4) which maps the posterior vector $p \in \mathbb{R}^{|\Sigma_{trg}|}$ to the w-th component.

The beam search strategy (Alg. 3) does not only keep the single best partial hypothesis but a set of n promising hypotheses where n is the size of the beam. A partial hypothesis is represented by a 3-tuple $(\mathbf{y}, p_{acc}, s)$ with the translation prefix $\mathbf{y} \in \Sigma_{trg}^*$, the accumulated score $p_{acc} \in \mathbb{R}$, and the last decoder state $s \in \mathbb{R}^n$.

Algorithm 3 BeamRNNsearch($s_{init}, \mathbf{h}, n \in \mathbb{N}_+$)

$\mathscr{H}_{cur} \leftarrow \{(\varepsilon, 0.0, s_{init})\}$ {Initialize with empty translation prefix and zero score}
repeat
  $\mathscr{H}_{next} \leftarrow \emptyset$
  for all $(\mathbf{y}, p_{acc}, s) \in \mathscr{H}_{cur}$ **do**
    if $y_{|\mathbf{y}|} =$ </s> **then**
      $\mathscr{H}_{next} \leftarrow \mathscr{H}_{next} \cup \{(\mathbf{y}, p_{acc}, s)\}$ {Hypotheses ending with </s> are not extended}
    else
      $s, p \leftarrow$ OneStepRNNsearch$(s, y_{|\mathbf{y}|}, \mathbf{h})$
      $\mathscr{H}_{next} \leftarrow \mathscr{H}_{next} \cup \bigcup_{w \in \Sigma_{trg}} (\mathbf{y} \cdot w, p_{acc} \pi_w(p), s)$ {Add all possible continuations}
    end if
  end for
  $\mathscr{H}_{cur} \leftarrow \{(\mathbf{y}, p_{acc}, s) \in \mathscr{H}_{next} : |\{(\mathbf{y}', p'_{acc}, s') \in \mathscr{H}_{next} : p'_{acc} > p_{acc}\}| < n\}$ {Select n-best}
  $(\hat{\mathbf{y}}, \hat{p}_{acc}, \hat{s}) \leftarrow \arg\max_{(\mathbf{y}, p_{acc}, s) \in \mathscr{H}_{cur}} p_{acc}$
until $\hat{y}_{|\hat{\mathbf{y}}|} =$ </s>
return $\hat{\mathbf{y}}$

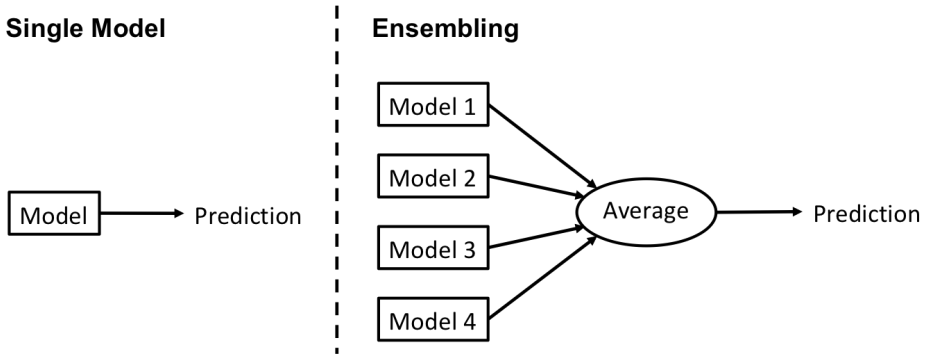


Fig. 3.17 Ensembling four NMT models.

3.7.4 Ensembling

Ensembling (Dietterich, 2000; Hansen and Salamon, 1990) of neural networks is a simple yet very effective technique to improve the accuracy of NMT. The basic idea is illustrated in Fig. 3.17. The decoder makes use of K NMT networks rather than only one which are either trained independently (Neubig, 2016; Sutskever et al., 2014; Wu et al., 2016b) or share some amount of training iterations (Cromieres et al., 2016; Durrani et al., 2016; Sennrich et al., 2016a). The ensemble decoder computes predictions for each of the individual models which are then combined using the arithmetic average (Sutskever et al., 2014) or the geometric

average (Cromieres et al., 2016).

$$S_{\text{arith}}(y_j|y_1^{j-1},\mathbf{x}) = \frac{1}{K}\sum_{k=1}^{K} P_k(y_j|y_1^{j-1},\mathbf{x}) \tag{3.23}$$

$$S_{\text{geo}}(y_j|y_1^{j-1},\mathbf{x}) = \sum_{k=1}^{K} \log P_k(y_j|y_1^{j-1},\mathbf{x}) \tag{3.24}$$

Both $S_{\text{arith}}(\cdot)$ and $S_{\text{geo}}(\cdot)$ can be used as drop-in replacement for the conditionals $P(y_j|y_1^{j-1},\mathbf{x})$ in Eq. 3.1. The arithmetic average is more sound as $S_{\text{arith}}(\cdot)$ still forms a valid probability distribution which sums up to one. However, the geometric average $S_{\text{arith}}(\cdot)$ is numerically more stable as log-probabilities can be directly combined without converting them to probabilities. Note that the core idea of ensembling is similar to language model interpolation used in statistical machine translation or speech recognition.

Ensembling consistently outperforms single NMT by a large margin. All top systems in recent machine translation evaluation campaigns ensemble a number of NMT systems (Bojar et al., 2017, 2016, 2019, 2018; Cromieres et al., 2016; Durrani et al., 2016; Junczys-Dowmunt, 2018b; Neubig, 2016; Sennrich et al., 2017a, 2016a; Stahlberg et al., 2018b; Wang et al., 2018b, 2017e), perhaps most famously taken to the extreme by the WMT18 submission of Tencent that ensembled up to 72 translation models (Wang et al., 2018b). However, the decoding speed is significantly worse since the decoder needs to apply K NMT models rather than only one. This means that the decoder has to perform K more forward passes through the networks, and has to apply the expensive softmax function K more times in each time step. Ensembling also often increases the number of CPU/GPU switches and the communication overhead between CPU and GPU when averaging is implemented on the CPU. Ensembling is also often more difficult to implement than single system NMT. Knowledge distillation which we will discuss in Sec. 3.16 is one method to deal with the shortcomings of ensembling. In Ch. 6, we will propose our own approach to efficient ensembling which is based on unfolding the ensemble into a single network and shrinking the unfolded network afterwards (Stahlberg and Byrne, 2017).

In NMT, all models in an ensemble usually have the same size and topology and are trained on the same data. They differ only due to the random weight initialization and the randomized order of the training samples. Notable exceptions include Freitag and Al-Onaizan (2016) who use ensembling to prevent overfitting in domain adaptation, He et al. (2018b) who combined models that selected their training data based on marginal likelihood, and our own submission

to WMT18 (Stahlberg et al., 2018b) (presented in Sec. 4.6) that ensembled different NMT architectures with each other.[8]

When all models are equally powerful and are trained with the same data, it is surprising that ensembling is so effective. One common narrative is that different models make different mistakes, but the mistake of one model can be outvoted by the others in the ensemble (Rokach, 2010). This explanation is plausible for NMT since translation quality can vary widely between training runs (Sennrich et al., 2016c). The variance in translation performance may also indicate that the NMT error surface is highly non-convex such that the optimizer often ends up in local optima. Ensembling might mitigate this problem. Ensembling may also have a regularization effect on the final translation scores (Goodfellow et al., 2016).

Checkpoint averaging (Junczys-Dowmunt et al., 2016a,b) is a technique which is often discussed in conjunction with ensembling (Liu et al., 2018c). Checkpoint averaging keeps track of the few most recent checkpoints during training, and averages their weight matrices to create the final model. This results in a single model and thus does not increase the decoding time. Therefore, it has become a very common technique in NMT (Popel and Bojar, 2018; Stahlberg et al., 2018b; Vaswani et al., 2017). However, checkpoint averaging addresses a quite different problem than ensembling as it mainly smooths out minor fluctuations in the training curve which are due to the optimizer's update rule or noise in the gradient estimation due to mini-batch training. In contrast, the weights of the (independently trained) models in an ensemble are very different from each other, and there is no obvious direct correspondence between neuron activity patterns across the models. Therefore, checkpoint averaging cannot be applied to independently trained models.

3.7.5 Decoding Direction

Standard NMT factorizes the probability $P(\mathbf{y}|\mathbf{x})$ from left to right (L2R) according Eq. 3.1. Mathematically, the left-to-right order is rather arbitrary, and other arrangements such as a right-to-left (R2L) factorization are equally correct:

$$P(\mathbf{y}|\mathbf{x}) = \underbrace{\prod_{j=1}^{J} P(y_j|y_1^{j-1},\mathbf{x})}_{=P(y_1|\mathbf{x})\cdot P(y_2|y_1,\mathbf{x})\cdot P(y_3|y_1,y_2,\mathbf{x})\cdots} = \underbrace{\prod_{j=1}^{J} P(y_j|y_{j+1}^{J},\mathbf{x})}_{=P(y_J|\mathbf{x})\cdot P(y_{J-1}|y_J,\mathbf{x})\cdot P(y_{J-2}|y_{J-1},y_J,\mathbf{x})\cdots} . \tag{3.25}$$

NMT models which produce the target sentence in reverse order have led to some gains in recent evaluation systems when combined with left-to-right models (Sennrich et al., 2016a;

[8] Multi-source ensembling (Firat et al., 2016b; Zoph and Knight, 2016) will be discussed in Sec. 3.15 in the context of multilingual NMT.

Stahlberg et al., 2018b; Wang et al., 2018b, 2017e). A common combination scheme is based on rescoring: A strong L2R ensemble first creates an n-best list which is then rescored with an R2L model (Liu et al., 2016a; Sennrich et al., 2016a). We will suggest an alternative way to use R2L models in Sec. 4.6 based on a minimum Bayes risk formulation. The L2R and R2L systems are normally trained independently, although some recent work proposes joint training schemes in which each direction is used as a regularizer for the other direction (Yang et al., 2018c; Zhang et al., 2018i). Other orderings besides L2R and R2L have also been proposed such as middle-out (Mehri and Sigal, 2018), top-down in a binary tree (Welleck et al., 2019), or insertion-based (Gu et al., 2019a,b; Östling and Tiedemann, 2017; Stern et al., 2019).

Another way to give the decoder access to the full target-side context is the two-stage approach of Li et al. (2017a) who first drafted a translation, and then employed a multi-source NMT system to generate the final translation from both the source sentence and the draft. Zhang et al. (2018e) proposed a similar scheme but generated the draft translations in reverse order. A similar two-pass approach was used by ElMaghraby and Rafea (2019) to make Arabic MT more robust against domain shifts. Geng et al. (2018) used reinforcement learning to choose the best number of decoding passes.

Besides explicit combination with an R2L model and multi-pass strategies, we are aware of following efforts to make the decoder more sensitive to the right-side target context: He et al. (2017) used reinforcement learning to estimate the long-term value of a candidate. Lin et al. (2018b) provided global target sentence information to a recurrent decoder via a convolutional model. Hoang et al. (2017) proposed a very appealing theoretical framework to relax the discrete NMT optimization problem into a continuous optimization problem which allows to include both decoding directions.

3.7.6 Efficiency

NMT decoding is very fast on GPU hardware and can reach up to 5000 words per second.[9] However, GPUs are very expensive, and speeding up CPU decoding to the level of SMT remains more challenging. Therefore, how to improve the efficiency of neural sequence decoding algorithms is still an active research question. One bottleneck is the sequential left-to-right order of beam search which makes parallelization difficult. Stern et al. (2018) suggested to compute multiple time steps in parallel and validate translation prefixes afterwards. Kaiser et al. (2018) reduced the amount of sequential computation by learning a sequence of latent discrete variables which is shorter than the actual target sentence, and generating the final sentence from this latent representation in parallel. Di Gangi and Federico (2018) sped up

[9] https://marian-nmt.github.io/features/

recurrent NMT by using a simplified architecture for recurrent units. Another line of research tries to reintroduce the idea of *hypothesis recombination* to neural models. This technique is used extensively in traditional SMT (Koehn, 2010). The idea is to keep only the better of two partial hypotheses if it is guaranteed that both will be scored equally in the future. For example, this is the case for n-gram language models if both hypotheses end with the same n-gram. The problem in neural sequence models is that they condition on the full translation history. Therefore, hypothesis recombination for neural sequence models does not insist on exact equivalence but cluster hypotheses based on the similarity between RNN states or the n-gram history (Liu et al., 2014; Zhang et al., 2018h). A similar idea was used by Lecorvé and Motlicek (2012) to approximate RNNs with WFSTs which also requires mapping histories into equivalence classes.

It is also possible to speed up beam search by reducing the beam size. Freitag and Al-Onaizan (2017); Wu et al. (2016b) suggested to use a variable beam size, using various heuristics to decide the beam size at each time step. Alternatively, the NMT model training can be tailored towards the decoding algorithm (Collobert et al., 2019; Goyal et al., 2018; Gu et al., 2017b; Wiseman and Rush, 2016). Wiseman and Rush (2016) proposed a loss function for NMT training which penalizes when the reference falls off the beam during training. Kim and Rush (2016) reported that knowledge distillation (discussed in Sec. 3.16) reduces the gap between greedy decoding and beam decoding significantly. Greedy decoding can also be improved by using a small actor network which modifies the hidden states in an already trained model (Chen et al., 2018d; Gu et al., 2017b).

3.7.7 Generating Diverse Translations

An issue with using beam search is that the hypotheses found by the decoder are very similar to each other and often differ only by one or two words (Gimpel et al., 2013; Li et al., 2016b; Li and Jurafsky, 2016). The lack of diversity is problematic for several reasons. First, natural language in general and translation in particular often come with a high level of ambiguity that is not represented well by non-diverse n-best lists. Second, it impedes user interaction as NMT is not able to provide the user with alternative translations if needed. Third, collecting statistics about the search space such as estimating the probabilities of n-grams for minimum Bayes-risk decoding (Sec. 4.5) or risk-based training (Sec. 3.11.5) is much less effective.

Cho (2016) added noise to the activations in the hidden layer of the decoder network to produce alternative high scoring hypotheses. This is justified by the observation that small variations of a hidden configuration encode semantically similar context (Bengio et al., 2013). Li et al. (2016b); Li and Jurafsky (2016) proposed a diversity promoting modification of the beam search objective function. They added an explicit penalization term to the NMT score

based on a maximum mutual information criterion which penalizes hypotheses from the same parent node. Note that both extensions can be used together (Cho, 2016). Vijayakumar et al. (2016) suggested to partition the active hypotheses in groups, and use a dissimilarity term to ensure diversity between groups. Park et al. (2016) found alternative translations by k-nearest neighbor search from the greedy translation in a translation memory.

3.7.8 Simultaneous Translation

Most of the research in MT assumes an offline scenario: a complete source sentence is to be translated to a complete target sentence. However, this basic assumption does not hold up for many real-life applications. For example, useful machine translation for parliamentary speeches and lectures (Fügen et al., 2007; Müller et al., 2016) or voice call services such as Skype (Lewis, 2015) does not only have to produce good translations but also have to do so with very low latency (Mieno et al., 2015). To reduce the latency in such real-time speech-to-speech translation scenarios it is desirable to start translating before the full source sentence has been vocalized by the speaker. Most approaches frame simultaneous machine translation as source sentence segmentation problem. The source sentence is revealed one word at a time. After a certain number of words, the segmentation policy decides to translate the current partial source sentence prefix and commit to a translation prefix which may not be a complete translation of the partial source. This process is repeated until the full source sentence is available. The segmentation policy can be heuristic (Cho and Esipova, 2016) or learned with reinforcement learning (Grissom II et al., 2014; Gu et al., 2017c). The translation itself is usually carried out by a standard MT system which was trained on full sentences. This is sub-optimal for two reasons. First, using a system which was trained on full sentences to translate partial sentences is brittle due to the significant mismatch between training and testing time. Ma et al. (2018b) tried to tackle this problem by training NMT to generate the target sentence with a fixed maximum latency to the source sentence. Second, human simultaneous interpreters use sophisticated strategies to reduce the latency by changing the grammatical structure (He et al., 2016b; Paulik and Waibel, 2009, 2013). These strategies are neglected by a vanilla translation system. Unfortunately, training data from human simultaneous translators is rare (Paulik and Waibel, 2013) which makes it difficult to adapt MT to it.

Vocabulary size	Number of parameters		
	Embeddings	Rest	Total
30K	55.8M	27.9M	83.7M
50K	93.1M	27.9M	121.0M
150K	279.2M	27.9M	307.1M

Table 3.3 Number of parameters in the original RNNsearch model (Bahdanau et al., 2015) as presented in Sec. 3.6.3 (1000 hidden units, 620-dimensional embeddings). The model size highly depends on the vocabulary size.

3.8 Open Vocabulary Neural Machine Translation

3.8.1 Using Large Output Vocabularies

As discussed in Sec. 3.2, NMT and other neural NLP models use embedding matrices to represent words as real-valued vectors. Embedding matrices need to have a fixed shape to make joint training with the translation model possible, and thus can only be used with a fixed and pre-defined vocabulary. This has several major implications for NMT.

First, the size of the embedding matrices grows with the vocabulary size. As shown in Tab. 3.3, the embedding matrices make up most of the model parameters of a standard RNNsearch model. Increasing the vocabulary size inflates the model drastically. Large models require a small batch size because they take more space in the (GPU) memory, but reducing the batch size often leads to noisier gradients, slower training, and eventually worse model performance (Popel and Bojar, 2018). Furthermore, a large softmax output layer is computationally very expensive. In contrast, traditional (symbolic) MT systems can easily use very large vocabularies (Chiang, 2007; Heafield et al., 2013; Koehn, 2010; Lin and Dyer, 2010).

Besides these practical issues, training embedding matrices for large vocabularies is also complicated by the long-tail distribution of words in a language. Zipf's law (Zipf, 1946) states that the frequency of any word and its rank in the frequency table are inversely proportional to each other. Fig. 3.18 shows that 843K of the 875K distinct words (96.5%) occur less than 100 times in an English text with 140M running words – that is less than 0.00007% of the entire text. It is difficult to train robust word embeddings for such rare words.

Word-based NMT models address this issue by restricting the vocabulary to the n most frequent words, and replacing all other words by a special token UNK. A problem with that approach is that the UNK token may appear in the generated translation. In fact, limiting the vocabulary to the 30K most frequent words results in an out-of-vocabulary rate (OOV) of 2.9% on the training set (Fig. 3.18). That means an UNK token can be expected to occur

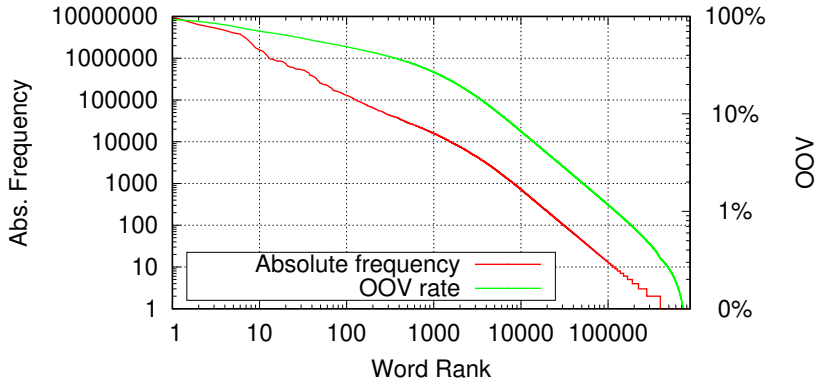


Fig. 3.18 Distribution of words in the English portion of the English-German WMT18 training set (5.9M sentences, 140M words).

every 35 words. In practice, the number of UNKs is usually even higher. One simple reason is that the test set OOV rate is often higher than on the training set because the distribution of words and phrases naturally varies across genre, corpora, and time. Another observation is that word-based NMT often prefers emitting UNK even if a more appropriate word is in the NMT vocabulary. This is possibly due to the misbalance between the UNK token and other words: replacing all rare words with the same UNK token leads to an over-representation of UNK in the training set, and therefore a strong bias towards UNK during decoding.

Translation-specific Approaches

Jean et al. (2015a) distinguished between *translation-specific* and *model-specific* approaches. Translation-specific approaches keep the shortlist vocabulary in the original form, but correct UNK tokens afterwards. For example, the UNK replace technique (Le et al., 2016; Luong et al., 2015c) keeps track of the positions of source sentence words which correspond to the UNK tokens. In a post-processing step, they replaced the UNK tokens with the most likely translation of the aligned source word according a bilingual word-level dictionary which was extracted from a word-aligned training corpus. Gulcehre et al. (2016) followed a similar idea but used a special pointer network for referring to source sentence words. These approaches are rather ad-hoc because simple dictionary lookup without context is not a very strong model of translation. Li et al. (2016c) replaced each OOV word with a similar in-vocabulary word based on the cosine similarity between their distributed representations in a pre-processing step. However, this technique cannot tackle all OOVs as it is based on vector representations of words which are normally only available for a closed vocabulary. Moreover, the replacements might differ from the original meaning significantly. Further UNK replacement strategies were presented by Li et al. (2017b,d); Miao et al. (2017), but all share the inevitable limitation of

all translation-specific approaches, namely that the translation model itself is indiscriminative between a large number of OOVs.

Model-specific Approaches

Model-specific approaches change the NMT model to make training with large vocabularies feasible. For example, Nguyen and Chiang (2018) improved the translation of rare words in NMT by adding a lexical translation model which directly connects corresponding source and target words. Another very popular idea is to train networks to output probability distributions without using the full softmax (Andreas and Klein, 2015). Noise-contrastive estimation (Dyer, 2014; Gutmann and Hyvärinen, 2010, NCE) trains a logistic regression model which discriminates between real training examples and noise. For example, to train an embedding for a word w, Mnih and Kavukcuoglu (2013) treat w as positive example, and sample from the global unigram word distribution in the training data to generate negative examples. The logistic regression model is a binary classifier and thus does not need to sum over the full vocabulary. NCE has been used to train large vocabulary neural sequence models such as language models (Mnih and Teh, 2012). The technique falls into the category of self-normalizing training (Andreas and Klein, 2015) because the model is trained to emit normalized distributions without explicitly summing over the output vocabulary. Self-normalization can also be achieved by adding the value of the partition function to the training loss (Devlin et al., 2014), encouraging the network to learn parameters which generate normalized output.

Another approach (sometimes referred to as *vocabulary selection*) is to approximate the partition function of the full softmax by using only a subset of the vocabulary. This subset can be selected in different ways. For example, Jean et al. (2015a) applied importance sampling to select a small set of words for approximating the partition function. Both softmax sampling and UNK replace have been used in one of the winning systems at the WMT'15 evaluation on English-German (Jean et al., 2015b). Various methods have been proposed to select the vocabulary to normalize over during decoding, such as fetching all possible translations in a conventional phrase table (Mi et al., 2016c), using the vocabulary of the translation lattices from a traditional MT system (Stahlberg et al., 2016b, local softmax), and attention-based (Sankaran et al., 2017) and embedding-based (L'Hostis et al., 2016) methods.

3.8.2 Character-based NMT

Arguably, both translation-specific and model-specific approaches to word-based NMT are fundamentally flawed. Translation-specific techniques like UNK replace are indiscriminative between translations that differ only by OOV words. A translation model which assigns

exactly the same score to a large number of hypotheses is of limited use by its own. Model-specific approaches suffer from the difficulty of training embeddings for rare words (Sec. 3.8.1). Compound or morpheme splitting (Hans and Milton, 2016; Tamchyna et al., 2017) can mitigate this issue only to a certain extent. More importantly, a fully-trained NMT system even with a very large vocabulary cannot be extended with new words. However, customizing systems to new domains (and thus new vocabularies) is a crucial requirement for commercial MT. Moreover, many OOV words are proper names which can be passed through untranslated. Hiero (Chiang, 2007) and other symbolic systems can easily be extended with new words and phrases.

More recent attempts try to alleviate the vocabulary issue in NMT by departing from words as modelling units. These approaches decompose the word sequences into finer-grained units and model the translation between those instead of words. To the best of our knowledge, Ling et al. (2015) were the first who proposed an NMT architecture which translates between sequences of characters. The core of their NMT network is still on the word-level, but the input and output embedding layers are replaced with subnetworks that compute word representations from the characters of the word. Such a subnetwork can be recurrent (Johansen et al., 2016; Ling et al., 2015) or convolutional (Costa-jussà and Fonollosa, 2016; Kim et al., 2016). This idea was extended to a hybrid model by Luong and Manning (2016) who used the standard lookup table embeddings for in-vocabulary words and the LSTM-based embeddings only for OOVs.

Having a word-level model at the core of a character-based system does circumvent the closed vocabulary restriction of purely word-based models, but it is still segmentation-dependent: The input text has to be preprocessed with a tokenizer that separates words by blank symbols in languages without word boundary markers, optionally applies compound or morpheme splitting in morphologically rich languages, and isolates punctuation symbols. Since tokenization is by itself error-prone and can degrade the translation performance (Domingo et al., 2018), it is desirable to design character-level systems that do not require any prior segmentation. Chung et al. (2016) used a bi-scale recurrent neural network that is similar to dynamically segmenting the input using jointly learned gates between a slow and a fast recurrent layer. Lee et al. (2017); Yang et al. (2016a) used convolution to achieve segmentation-free character-level NMT. Costa-jussà et al. (2017) took character-level NMT one step further and used bytes rather than characters to help multilingual systems. Gulcehre et al. (2017a) added a planning mechanism to improve the attention weights between character-based encoders and decoders.

3.8.3 Subword-unit-based NMT

As compromise between characters and full words, compression methods like Huffman codes (Chitnis and DeNero, 2015), word piece models (Schuster and Nakajima, 2012; Wu et al., 2016b), or byte pair encoding (Gage, 1994; Sennrich et al., 2016c, BPE) can be used to transform the words to sequences of subword units. Subwords have been used rarely for traditional SMT (Kunchukuttan and Bhattacharyya, 2016, 2017; Liu et al., 2018a), but are currently the most common translation units for NMT. We will therefore discuss one particularly popular subword model (BPE) in greater detail.

Byte pair encoding (BPE) initializes the set of available subword units with the character set of the language. This set is extended iteratively in subsequent merge operations. Each merge combines the two units with the highest number of co-occurrences in the text.[10] This process terminates when the desired vocabulary size is reached. This vocabulary size is often set empirically, but can also be tuned on data (Salesky et al., 2018). Fig. 3.19 illustrates the algorithm. The two most common letter combinations in the text or "ou" and "ha", so they are replaced by new symbols 'A' and 'B' in the first two iterations. The next merge operation combines the newly introduced symbol 'A' with yet another character ('t'), making 'C' a subword unit consisting of three characters. BPE is sometimes described as bottom-up hierarchical clustering algorithm because rearranging the merge operations like in Fig. 3.20 resembles a tree in the first few iterations. However, the analogy is limited since the structure in Fig. 3.20 is generally not a tree, e.g. if 'h' and 'D' are eventually merged by the algorithm.

Given a fixed BPE vocabulary, there are often multiple ways to segment an unseen text.[11] The ambiguity stems from the fact that symbols are still part of the vocabulary even after they are merged. Most BPE implementations select a segmentation greedily by preferring longer subword units. Interestingly, the ambiguity can also be used as source of noise for regularization. Kudo (2018) reported surprisingly large gains by augmenting the training data with alternative subword segmentations and by decoding from multiple segmentations of the same source sentence.

Segmentation approaches differ in the level of constraints they impose on the subwords. A common constraint is that subwords cannot span over multiple words (Sennrich et al., 2016c). However, enforcing this constraint again requires a tokenizer which is a potential source of errors (see Sec. 3.8.2). The SentencePiece model (Kudo and Richardson, 2018) is a tokenization-free subword model that is estimated on raw text. On the other side of the

[10] Wu and Zhao (2018) proposed alternatives to the co-occurrence counts. The wordpiece model (Schuster and Nakajima, 2012; Wu et al., 2016b) can also be seen as replacing the co-occurrence counts with a language model objective.

[11] This is not true for other subword compression algorithms. For example, Huffman codes (Chitnis and DeNero, 2015) are prefix codes and thus unique.

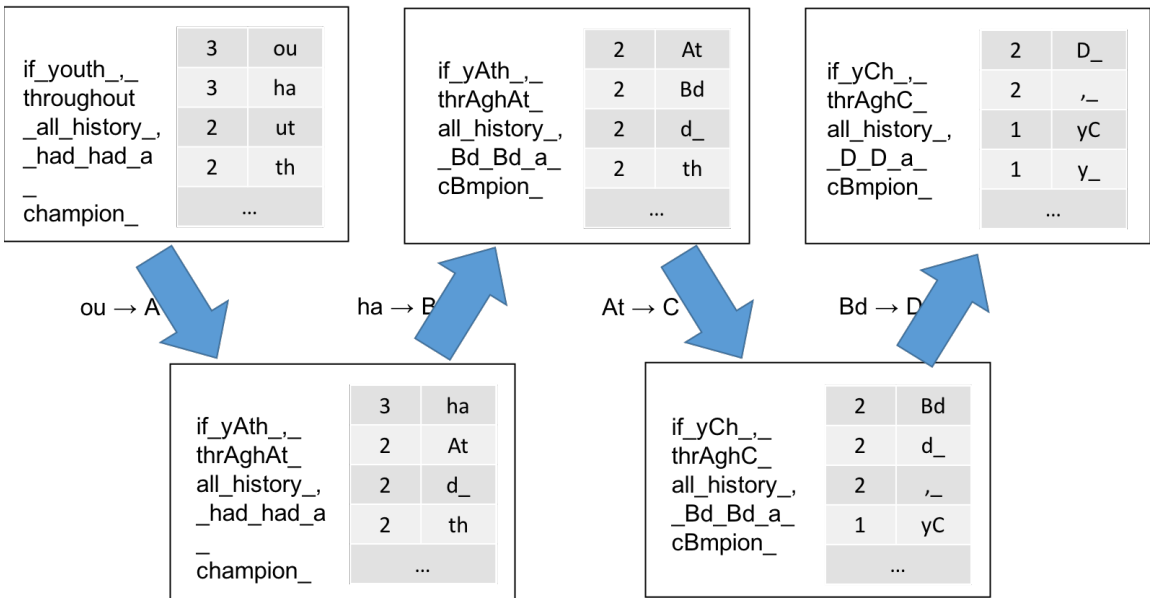


Fig. 3.19 First four merge operations of the byte pair encoding (BPE) algorithm on the text "if youth, throughout all history, had had a champion" (the opening phrase of the 260 page novel *Gadsby* from EV Wright which deliberately contains only words without the letter 'E'). Counts indicate the frequency of symbol bigrams in the text.

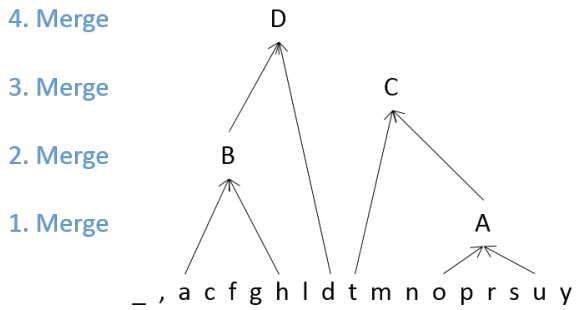


Fig. 3.20 Hierarchical representation of the merge operations in the example in Fig. 3.19.

spectrum, it has been observed that automatically learned subwords generally do not correspond to linguistic entities such as morphemes, suffixes, affixes etc. However, linguistically-motivated subword units (Ataman et al., 2017; Huck et al., 2017; Macháček et al., 2018; Pinnis et al., 2017) that also take morpheme boundaries into account do not always improve over completely data-driven ones.

Character-based NMT	Subword-based NMT
+ Better at transliteration (Sennrich, 2017).	+ More grammatical (Sennrich, 2017).
+ Dynamic segmentation favors characters (Kreutzer and Sokolov, 2018).	+ Iterative BPE segmentation favors larger vocabulary sizes (Salesky et al., 2018).
+ More robust against noise (Belinkov and Bisk, 2017; Durrani et al., 2018).	+ Better at syntax (Durrani et al., 2018).
+ Better modelling of morphology (Durrani et al., 2018).	+ Tends to outperform character-based models in recent MT evaluations (Bojar et al., 2017, 2016, 2018).
+ Character-level decoders better than subword-based ones in some studies (Cherry et al., 2018; Chung et al., 2016).	
− Character-based NMT computationally more expensive than subword-based NMT (Cherry et al., 2018).	
− More prone to vanishing gradients (Chung et al., 2016).	
− Long-range dependencies have to be modelled over longer timespans (Lee et al., 2017).	

Table 3.4 Summary of studies comparing characters and subword-units for neural machine translation.

3.8.4 Words, Subwords, or Characters?

There is no conclusive agreement in the literature whether characters or subwords are the better translation units for NMT. Tab. 3.4 summarizes some of the arguments. The tendency seems to be that character-based systems have the potential of outperforming subword-based NMT, but they are technically difficult to deploy. Therefore, most systems in the WMT18 evaluation are based on subwords (Bojar et al., 2018).

On a more profound level, we do see the shift towards small modelling units not without some concern. Chung et al. (2016) noted that "we often have a priori belief that a word, or its segmented-out lexeme, is a basic unit of meaning, making it natural to approach translation as mapping from a sequence of source-language words to a sequence of target-language words." Translation is the task of transferring *meaning* from one language to another, and it makes intuitive sense to model this process with meaningful units. The decades of research in traditional SMT were characterized by a constant movement towards larger translation units – starting from the word-based IBM models (Brown et al., 1993) to phrase-based MT (Koehn, 2010) and hierarchical SMT (Chiang, 2007) that models syntactic structures. Expressions consisting of multiple words are even more appropriate units than words for translation since there is rarely a 1:1 correspondence between source and target words. In contrast, the starting point for character- and subword-based models is the language's writing system. Most writing systems are not logographic but alphabetic or syllabaric and thus use symbols without any relation to meaning. The introduction of symbolic word-level and phrase-level information to NMT is one of our main reasons for studying NMT-SMT hybrid systems in Ch. 4.

3.9 Using Monolingual Training Data

In practice, parallel training data for MT is hard to acquire and expensive, whereas untranslated monolingual data is usually abundant. This is one of the reasons why language models (LMs) are central to traditional SMT. For example, in Hiero (Chiang, 2007), the translation grammar spans a vast space of possible translations but is weak in assigning scores to them. The LM is mainly responsible for selecting a coherent and fluent translation from that space. However, the vanilla NMT formalism does not allow the integration of an LM or monolingual data in general.

There are several lines of research which investigate the use of monolingual training data in NMT. Gulcehre et al. (2015, 2017b) suggested to integrate a separately trained RNN-LM into the NMT decoder. Similarly to traditional SMT (Koehn, 2010) they started out with combining RNN-LM and NMT scores via a log-linear model ('shallow fusion'). They reported even better performance with 'deep fusion' which uses a controller network that dynamically adjusts the weights between RNN-LM and NMT. Both deep fusion and n-best reranking with count-based language models have led to some gains in WMT evaluation systems (Jean et al., 2015b; Wang et al., 2017e). We will explore the use of language models for neural sequence-to-sequence prediction extensively in Ch. 7.

The second line of research makes use of monolingual text via data augmentation. The idea is to add monolingual data in the target language to the natural parallel training corpus. Different

strategies for filling in the source side for these sentences have been proposed such as using a single dummy token (Sennrich et al., 2016b) or copying the target sentence over to the source side (Currey et al., 2017). The most successful strategy is called back-translation (Schwenk, 2008; Sennrich et al., 2016b) which employs a separate translation system in the reverse direction to generate the source sentences for the monolingual target language sentences. The back-translating system is usually smaller and computationally cheaper than the final system for practical reasons, although with enough computational resources improving the quality of the reverse system can affect the final translation performance significantly (Burlot and Yvon, 2018). Iterative approaches that back-translate with systems that were by themselves trained with back-translation can yield improvements (Hoang et al., 2018b; Niu et al., 2018; Zhang et al., 2018g) although they are not widely used due to their computational costs. Back-translation has become a very common technique and has been used in nearly all neural submissions to recent evaluation campaigns (Bojar et al., 2017, 2018; Sennrich et al., 2016a).

A major limitation of back-translation is that the amount of synthetic data has to be balanced with the amount of real parallel data (Poncelas et al., 2018; Sennrich et al., 2016a,b). Therefore, the back-translation technique can only make use of a small fraction of the available monolingual data. A misbalance between synthetic and real data can be partially corrected by over-sampling – duplicating real training samples a number of times to match the synthetic data size. However, very high over-sampling rates often do not work well in practice. Recently, Edunov et al. (2018a) proposed to add noise to the back-translated sentences to provide a stronger training signal from the synthetic sentence pairs. They showed that adding noise does not only improve the translation quality but also makes the training more robust against a high ratio of synthetic against real sentences. The effectiveness of using noise for data augmentation in NMT has also been confirmed by Wang et al. (2018e). These methods increase the variety of the training data and thus make it harder for the model to fit which ultimately leads to stronger training signals. The variety of synthetic sentences in back-translation can also be increased by sampling multiple sentences from the reverse translation model (Imamura et al., 2018).

A third class of approaches changes the NMT training loss function to incorporate monolingual data. For example, Cheng et al. (2016c); Escolano et al. (2018); Tu et al. (2017) proposed to add autoencoder terms to the training objective which capture how well a sentence can be reconstructed from its translated representation. Using the reconstruction error is also central to (unsupervised) dual learning approaches (Hassan et al., 2018; He et al., 2016a; Wang et al., 2018i). However, training with respect to the new loss is often computationally intensive and requires approximations. Alternatively, multi-task learning has been used to incorporate source-side (Zhang and Zong, 2016b) and target-side (Domhan and Hieber, 2017) monolingual data. Another way of utilizing monolingual data in both source and target language is to warm

start Seq2Seq training from pre-trained encoder and decoder networks (Ramachandran et al., 2017; Skorokhodov et al., 2018).

An extreme form of leveraging monolingual training data is unsupervised NMT which removes the need for parallel training data entirely. We will discuss unsupervised NMT in Sec. 3.14.4.

3.10 NMT Model Errors

NMT is highly effective in assigning scores (or probabilities) to translations because, in stark contrast to SMT, it does not make any conditional independence assumptions in Eq. 3.1 to model sentence-level translation.[12] A potential drawback of such a powerful model is that it prohibits the use of sophisticated search procedures. Compared to hierarchical SMT systems like Hiero (Sec. 2.7.2) that explore very large search spaces, NMT beam search appears to be overly simplistic. This observation suggests that translation errors in NMT are more likely due to *search errors* (the decoder does not find the highest scoring translation) than *model errors* (the model assigns a higher probability to a worse translation). Interestingly, this is not necessarily the case. Besides some preliminary studies (Niehues et al., 2017; Stahlberg et al., 2018d), analyzing search errors in NMT has received only limited attention, probably because the large size of the NMT search space makes it difficult to find oracle scores.[13] However, as we will show in the next sections, NMT does suffer from various kinds of model errors in practice despite its theoretical advantage.

3.10.1 Sentence Length

Increasing the beam size exposes one of the most noticeable model errors in NMT. The red curve in Fig. 3.21 plots the BLEU score (Papineni et al., 2002) of a recent Transformer NMT model against the beam size. A beam size of 10 is optimal on this test set. Wider beams lead to a steady drop in translation performance because the generated translations are becoming too short (green curve). However, as expected, the log-probabilities of the found translations (blue curve) are decreasing as we increase the beam size. NMT seems to assign too much probability mass to short hypotheses which are only found with more exhaustive search. Sountsov and Sarawagi (2016) argue that this model error is due to the locally normalized maximum likelihood training objective in NMT that underestimates the margin between the correct translation and shorter

[12] It does, however, assume that each sentence can be translated in isolation. We will take a closer look at this assumption in Sec. 3.17.4.

[13] In Sec. 5.4.4 we present an algorithm that enables us to study NMT search errors and model errors despite the large NMT search space.

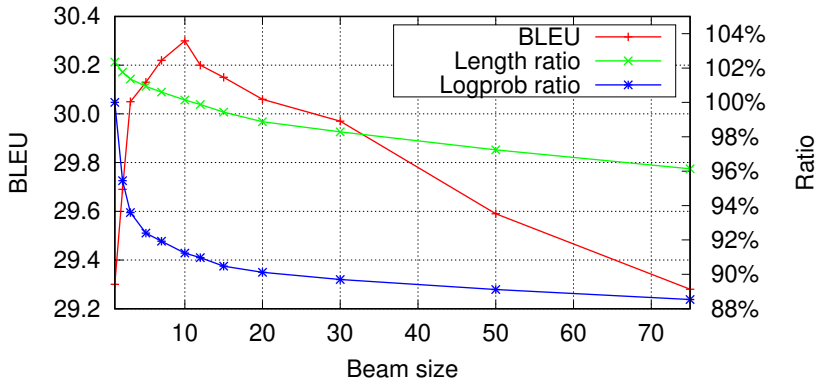


Fig. 3.21 Performance of a Transformer model on English-German (WMT15) under varying beam sizes. The BLEU score peaks at beam size 10, but then suffers from a length ratio (hypothesis length / reference length) below 1. The log-probabilities are shown as a ratio with respect to greedy decoding.

ones if trained with regularization and finite data. A similar argument was made by Murray and Chiang (2018) who pointed out the difficulty for a locally normalized model to estimate the "budget" for all remaining (longer) translations in each time step. Kumar and Sarawagi (2019) demonstrated that NMT models are often poorly calibrated, and that calibration issues can cause the length deficiency in NMT. A similar case is illustrated in Fig. 3.22. The NMT model underestimates the combined probability mass of translations continuing after "Stadtrat" in time step 7 and overestimates the probability of the period symbol. Greedy decoding does not follow the green translation since "der" is more likely in time step 7. However, beam search with a large beam keeps the green path and thus finds the shorter (incomplete) translation with better score.

At first glance this seems to be good news: fast beam search with a small beam size is already able to find good translations. However, fixing the model error of short translations by introducing search errors with a narrow beam seems like fighting fire with fire. In practice, this means that the beam size is yet another hyper-parameter which needs to be tuned for each new NMT training technique (eg. label smoothing (Szegedy et al., 2016) usually requires a larger beam), NMT architecture (the Transformer model is usually decoded with a smaller beam than typical recurrent models), and language pair (Koehn and Knowles, 2017). More importantly, it is not clear whether there are gains to be had from reducing the number of search errors with wider beams which are simply obliterated by the NMT length deficiency.

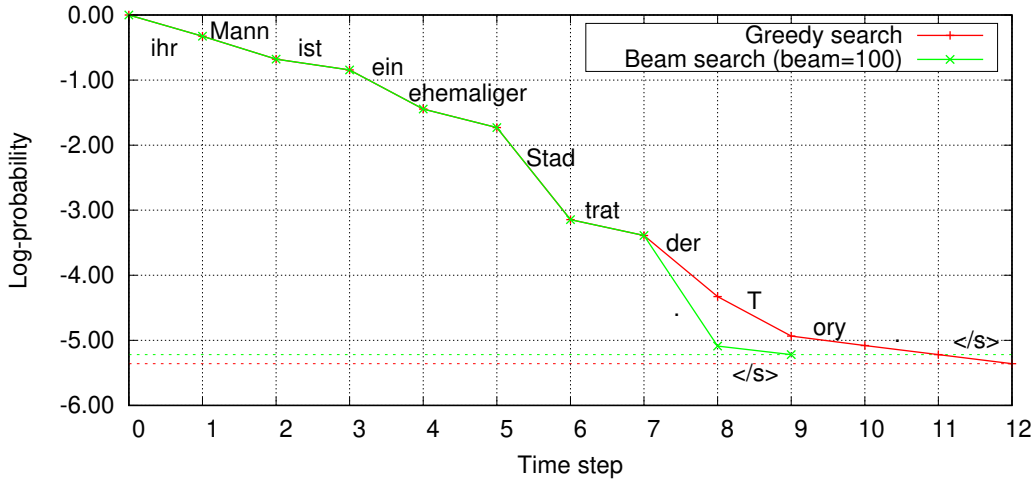


Fig. 3.22 The length deficiency in NMT translating the English source sentence "Her husband is a former Tory councillor." into German following Murray and Chiang (2018). The NMT model assigns a better score to the short translation "Ihr Mann ist ein ehemaliger Stadtrat." than to the greedy translation "Ihr Mann ist ein ehemaliger Stadtrat der Tory." even though it misses the former affiliation of the husband with the Tory Party.

Model-agnostic Length Models

The first class of approaches to alleviate the length problem is model-agnostic. Methods in this class treat the NMT model as black box but add a correction term to the NMT score to bias beam search towards longer translations. A simple method is called *length normalization* which divides the NMT probability by the sentence length (Boulanger-Lewandowski et al., 2013; Jean et al., 2015b):

$$S_{\text{LN}}(\mathbf{y}|\mathbf{x}) = \frac{\log P(\mathbf{y}|\mathbf{x})}{|\mathbf{y}|} \tag{3.26}$$

Wu et al. (2016b) proposed an extension of this idea by introducing a tunable parameter α:

$$S_{\text{LN-GNMT}}(\mathbf{y}|\mathbf{x}) = \log P(\mathbf{y}|\mathbf{x}) \frac{(1+5)^{\alpha}}{(1+|\mathbf{y}|)^{\alpha}} \tag{3.27}$$

Alternatively, like in SMT we can use a word penalty $\gamma(j,\mathbf{x})$ which rewards each word in the sentence:

$$S_{\text{WP}}(\mathbf{y}|\mathbf{x}) = \sum_{j=1}^{J} \gamma(j,\mathbf{x}) + \log P(y_j|y_1^{j-1},\mathbf{x}) \tag{3.28}$$

A constant reward which is independent of $\mathbf{x}$ and j can be found with the standard minimum-error-rate-training (Och, 2003, MERT) algorithm (He et al., 2016d) or with a gradient-based learning scheme (Murray and Chiang, 2018). Alternative policies which reward words with

respect to some estimated sentence length were suggested by Huang et al. (2017a); Yang et al. (2018b).

Source-side Coverage Models

Tu et al. (2016) connected the sentence length issue in NMT with the lack of an explicit mechanism to check the source-side coverage of a translation. Traditional SMT keeps track of a coverage vector $\mathscr{C}_{\text{SMT}} \in \{0,1\}^I$ which contains 1 for source words which are already translated and 0 otherwise. $\mathscr{C}_{\text{SMT}}$ is used to guard against *under-translation* (missing translations of some words) and *over-translation* (some words are unnecessarily translated multiple times). Since vanilla NMT does not use an explicit coverage vector it can be prone to both under- and over-translation (Tu et al., 2016; Yang et al., 2018a) and tends to prefer fluency over adequacy (Kong et al., 2018). There are two popular ways to model coverage in NMT, both make use of the encoder-decoder attention weight matrix A introduced in Sec. 3.6.1. The simpler methods combine the scores of an already trained NMT system with a coverage penalty $cp(\mathbf{x},\mathbf{y})$ without retraining. This penalty represents how much of the source sentence is already translated. Wu et al. (2016b) proposed the following term:

$$cp(\mathbf{x},\mathbf{y}) = \beta \sum_{i=1}^{I} \log\big(\min(\sum_{j=1}^{J} A_{i,j}, 1.0)\big). \tag{3.29}$$

A very similar penalty was suggested by Li et al. (2018):

$$cp(\mathbf{x},\mathbf{y}) = \alpha \sum_{i=1}^{I} \log\big(\max(\sum_{j=1}^{J} A_{i,j}, \beta)\big) \tag{3.30}$$

where α and β are hyper-parameters that are tuned on the development set.

An even tighter integration can be achieved by changing the NMT architecture itself and jointly training it with a coverage model (Mi et al., 2016a; Tu et al., 2016). Tu et al. (2016) reintroduced an explicit coverage matrix $\mathscr{C} \in [0,1]^{I \times J}$ to NMT. Intuitively, the j-th column $\mathscr{C}_{:,j}$ stores to what extend each source word has been translated in time step j. $\mathscr{C}$ can be filled with an RNN-based controller network (the "neural network based" coverage model of Tu et al. (2016)). Alternatively, we can directly use A to compute the coverage (the "linguistic" coverage model of Tu et al. (2016)):

$$\mathscr{C}_{i,j} = \frac{1}{\Phi_i} \sum_{k=1}^{j} A_{i,k} \tag{3.31}$$

where Φ_i is the estimated number of target words the i-th source word generates which is similar to fertility in SMT (Sec. 2.4). Φ_i is predicted by a feedforward network that conditions

on the i-th encoder state. In both the neural network based and the linguistic coverage model, the decoder is modified to additionally condition on $\mathscr{C}$. The idea of using fertilities to prevent over- and under-translation has also been explored by Malaviya et al. (2018). A coverage model for character-based NMT was suggested by Kazimi and Costa-Jussá (2017).

All approaches discussed in this section operate on the attention weight matrix A and are thus only readily applicable to models with single encoder-decoder attention like GNMT, but not to models with multiple encoder-decoder attention modules such as ConvS2S or the Transformer (see Sec. 3.6.6 for detailed descriptions of GNMT, ConvS2S, and the Transformer).

Controlling Mechanisms for Output Length

In some sequence prediction tasks such as headline generation or text summarization, the approximate desired output length is known in advance. In such cases, it is possible to control the length of the output sequence by explicitly feeding in the desired length to the neural model. The length information can be provided as additional input to the decoder network (Fan et al., 2018; Liu et al., 2018b), at each time step as the number of remaining tokens (Kikuchi et al., 2016), or by modifying Transformer positional embeddings (Takase and Okazaki, 2019). However, these approaches are not directly applicable to machine translation as the translation length is difficult to predict with sufficient accuracy.

3.11 NMT Training

NMT models are normally trained using backpropagation (Rumelhart et al., 1988) and a gradient-based optimizer like Adadelta (Zeiler, 2012) with cross-entropy loss (Sec. 3.11.1). Modern NMT architectures like the Transformer, ConvS2S, or recurrent networks with LSTM (Hochreiter and Schmidhuber, 1997) or GRU (Cho et al., 2014b) cells help to address known training problems like vanishing gradients (Hochreiter et al., 2001). However, there is evidence that the optimizer still fails to exploit the full potential of NMT models and often gets stuck in suboptima:

1. NMT models vary greatly in performance, even if they use exactly the same architecture, training data, and are trained for the same number of iterations. Sennrich et al. (2016c) observed up to 1 BLEU difference between different models.

2. NMT ensembling (Sec. 3.17) combines the scores of multiple separately trained NMT models of the same kind. NMT ensembles consistently outperform single NMT by

a large margin. The achieved gains through ensembling might indicate difficulties in training of the single models.[14]

Training is therefore still a very active and diverse research topic. We will outline the different efforts in the literature on NMT training in this section.

3.11.1 Cross-entropy Training

The most common objective function for NMT training is cross-entropy loss. The optimization problem over model parameters Θ for a single sentence pair $(\mathbf{x}, \mathbf{y})$ under this loss is defined as follows:

$$\arg\min_{\Theta} \mathscr{L}_{\text{CE}}(\mathbf{x}, \mathbf{y}, \Theta) = \arg\min_{\Theta} -\sum_{j=1}^{|\mathbf{y}|} \log P_{\Theta}(y_j | y_1^{j-1}, \mathbf{x}). \tag{3.32}$$

In practice, NMT training groups several instances from the training corpus into batches, and optimizes Θ by following the gradient of the average $\mathscr{L}_{\text{CE}}(\mathbf{x}, \mathbf{y}, \Theta)$ in the batch. There are various ways to interpret this loss function.

Cross-entropy loss maximizes the log-likelihood of the training data A direct interpretation of Eq. 3.32 is that it yields a maximum likelihood estimate of Θ as it directly maximizes the probability $P_{\Theta}(\mathbf{y}|\mathbf{x})$:

$$\log P_{\Theta}(\mathbf{y}|\mathbf{x}) \overset{\text{Eq. 3.1}}{=} -\sum_{j=1}^{|\mathbf{y}|} \log P_{\Theta}(y_j | y_1^{j-1}, \mathbf{x}) = \mathscr{L}_{\text{CE}}(\mathbf{x}, \mathbf{y}, \Theta). \tag{3.33}$$

Cross-entropy loss optimizes a Monte Carlo approximation of the cross-entropy to the real sequence-level distribution Another intuition behind the cross-entropy loss is that we want to find model parameters Θ that make the model distribution $P_{\Theta}(\cdot|\mathbf{x})$ similar to the *real* distribution $P(\cdot|\mathbf{x})$ over translations for a source sentence $\mathbf{x}$. The similarity is measured with the cross-entropy $H_{\mathbf{x}}(P, P_{\Theta})$. In practice, the real distribution $P(\cdot|\mathbf{x})$ is not known, but we have access to a training corpus of pairs $(\mathbf{x}, \mathbf{y})$. For each such pair we consider the target sentence $\mathbf{y}$ as a *sample* from the real distribution $P(\cdot|\mathbf{x})$. We now approximate the cross-entropy $H_{\mathbf{x}}(P, P_{\theta})$

[14] I thank Adrià de Gispert for making that point in our discussions.

using Monte Carlo estimation with only one sample ($N = 1$):

$$\begin{aligned} H_{\mathbf{x}}(P, P_\Theta) &= \mathbb{E}_{\mathbf{y}}[-\log P_\Theta(\mathbf{y}|\mathbf{x})] \\ &= -\sum_{\mathbf{y}'} P(\mathbf{y}'|\mathbf{x}) \log P_\Theta(\mathbf{y}'|\mathbf{x}) \\ &\overset{\text{MC}}{\approx} -\frac{1}{N}\sum_{\mathbf{y}'} \log P_\Theta(\mathbf{y}'|\mathbf{x}) \\ &\overset{N=1}{=} -\log P_\Theta(\mathbf{y}|\mathbf{x}) \\ &= -\sum_{j=1}^{|\mathbf{y}|} \log P_\Theta(y_j|y_1^{j-1}, \mathbf{x}) \\ &= \mathscr{L}_{\text{CE}}(\mathbf{x}, \mathbf{y}, \Theta). \end{aligned}$$

Cross-entropy loss optimizes a Monte Carlo approximation of the cross-entropy to the real token-level distribution We arrive at the same result if we consider the cross-entropy between the *conditionals* of $P(\cdot|y_1^{j-1}, \mathbf{x})$ and $P_\Theta(\cdot|y_1^{j-1}, \mathbf{x})$ for given $\mathbf{x}$ and translation prefix y_1^{j-1}:

$$\begin{aligned} \mathbb{E}_{y_j}[-\log P_\Theta(y_j|y_1^{j-1}, \mathbf{x})] &= -\sum_{y'_j} P(y'_j|y_1^{j-1}, \mathbf{x}) \log P_\Theta(y'_j|y_1^{j-1}, \mathbf{x}) \\ &\overset{\text{MC with } N=1}{\approx} -\sum_{j=1}^{|\mathbf{y}|} \log P_\Theta(y_j|y_1^{j-1}, \mathbf{x}) \\ &= \mathscr{L}_{\text{CE}}(\mathbf{x}, \mathbf{y}, \Theta). \end{aligned}$$

Cross-entropy loss optimizes the cross-entropy to the Dirac distribution Alternatively, we can define a (Dirac) distribution which assigns the probability of one to $\mathbf{y}$ and zero to all other target sentences:

$$P_\delta(\mathbf{y}'|\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{y}' = \mathbf{y} \\ 0 & \text{if } \mathbf{y}' \neq \mathbf{y} \end{cases} \tag{3.34}$$

The cross-entropy between the Dirac distribution (in this context taking the role of the empirical distribution) and our model distribution $P_\Theta(\cdot|\mathbf{x})$ is:

$$H_{\mathbf{x}}(P_\delta, P_\Theta) = -\sum_{\mathbf{y}'} P_\delta(\mathbf{y}'|\mathbf{x}) \log P_\Theta(\mathbf{y}'|\mathbf{x}) = -\log P_\Theta(\mathbf{y}|\mathbf{x}) \overset{\text{Eq. 3.33}}{=} \mathscr{L}_{\text{CE}}(\mathbf{x}, \mathbf{y}, \Theta). \tag{3.35}$$

To recap, we have found that the following are equivalent:

- Training under cross-entropy loss (Eq. 3.32).

- Maximizing the likelihood of the training data.
- Minimizing an estimate of the cross-entropy to the real sequence-level distribution.
- Minimizing an estimate of the cross-entropy to the real token-level distribution.
- Minimizing the cross-entropy to the Dirac distribution.

In particular, we emphasize the equivalence between the sequence-level and the token-level estimation since cross-entropy loss is often characterized as token-level objective in the literature whereas the term *sequence-level training* somewhat misleadingly usually refers to risk-based training under BLEU (Edunov et al., 2018b; Ranzato et al., 2015) which is discussed in Sec. 3.11.5.

3.11.2 Training Deep Architectures

Deep encoders and decoders consisting of multiple layers have now superseded earlier shallow architectures. However, since the gradients have to be back-propagated through more layers, deep architectures – especially recurrent ones – are prone to vanishing gradients (Pascanu et al., 2013) and are thus harder to train. A number of tricks have been proposed recently that make it possible to train deep NMT models reliably. Residual connections (He et al., 2016c) are direct connections that bypass more complex sub-networks in the layer stack. For example, all the architectures presented in Sec. 3.6.6 (GNMT, ConvS2S, Transformer, RNMT+) add residual connections around attentional, recurrent, or convolutional cells to ease learning (Fig. 3.15). Another technique to counter vanishing gradients is called *batch normalization* (Ioffe and Szegedy, 2015) which normalizes the hidden activations in each layer in a mini-batch to a mean of zero and a variance of 1. An extension of batch normalization which is independent of the batch size and is especially suitable for recurrent networks is called *layer normalization* (Ba et al., 2016). Layer normalization is popular for training deep NLP models like the Transformer (Vaswani et al., 2017).

3.11.3 Regularization

Modern NMT architectures are vastly over-parameterized (Stahlberg and Byrne, 2017) to help training (Livni et al., 2014). For example, a subword-unit-level Transformer in a standard "big" configuration can easily have 200-300 million parameters (Stahlberg et al., 2018b). The large number of parameters potentially makes the model prone to *over-fitting*: The model fits the training data perfectly, but the performance on held-out data suffers as the large number of parameters allows the optimizer to marginally improve training loss at the cost of generalization

as training proceeds. Techniques that aim to prevent over-fitting in over-parameterized neural networks are called *regularizers*. Perhaps the two simplest regularization techniques are L1 and L2 regularization. The idea is to add terms to the loss function that penalize the magnitude of weights in the network. Intuitively, such penalties draw many parameters towards zero and limit their significance. Thus, L1 and L2 effectively serve as soft constraint on the model capacity.

The three most popular regularization techniques for NMT are *early stopping*, *dropout*, and *label smoothing*. Early stopping can be seen as regularization in time as it stops training as soon as the performance on the development set does not improve anymore. Dropout (Srivastava et al., 2014) is arguably one of the key techniques that have made deep learning practical. Dropout randomly sets the activities of hidden and visible units to zero during training. Thus, it can be seen as a strong regularizer for simultaneously training a large collection of networks with extensive weight sharing.

Label smoothing has been derived for expectation–maximization training by Byrne (1993), and has been applied to large-scale computer vision by Szegedy et al. (2016). Label smoothing changes the training objective such that the model produces smoother distributions. We have already established in Sec. 3.11.1 that standard cross-entropy training measures the distance of the output distribution to the Dirac distribution around the training sample. Label smoothing discounts the likelihood of the training sample and distributes some of the free probability mass among other hypotheses. In NMT, label smoothing is applied as cross-entropy loss to a smoothed distribution $Q(\cdot)$ on the token level:

$$\mathscr{L}_Q(\mathbf{x},\mathbf{y},\Theta) = -\sum_{j=1}^{|\mathbf{y}|}\sum_{y'\in\Sigma_{trg}} Q(y'|j,\mathbf{y})\log P_\Theta(y'|y_1^{j-1},\mathbf{x}). \tag{3.36}$$

The distribution $Q(\cdot)$ can take language modelling scores into account (Chorowski and Jaitly, 2017), but usually it is just a smoothed version of the Dirac distribution for the reference label:

$$Q_\alpha(y'|j,\mathbf{y}) = \begin{cases} \alpha & \text{if } y' = \mathbf{y}_j \\ \frac{1-\alpha}{|\Sigma_{trg}|-1} & \text{if } y' \neq \mathbf{y}_j \end{cases} \tag{3.37}$$

for some smoothing factor $\alpha \in (0,1]$. Setting $\alpha = 1$ recovers the normal cross-entropy loss from Sec. 3.11.1.

While label smoothing makes intuitive sense for computer vision, applying it to neural sequence prediction in this way has objectionable side effects on the sequence level. Considering the probabilities $Q(\cdot)$ assigns to *full sequences*, we first note that $Q(\cdot)$ does *not* uniformly distribute the remaining probability mass among all other sequences. In fact, distributing it uniformly would result in infinitely small probabilities as there are infinitely many possible

sequences. Interestingly, $Q(\cdot)$ does also not assign a fixed probability of α to the correct sequence $\mathbf{y}$:

$$Q_\alpha(\mathbf{y}) = \prod_{j=1}^{|\mathbf{y}|} Q_\alpha(y_j|y_1^{j-1}) = \prod_{j=1}^{|\mathbf{y}|} \alpha = \alpha^{|\mathbf{y}|}. \tag{3.38}$$

Since α is less than one, $Q_\alpha(\cdot)$ is sharper if the correct sequence $\mathbf{y}$ is short, and smoother if it is long. In theory, this should bias NMT training toward producing shorter translations. One way to fix this bias would be to smooth with $\sqrt[|\mathbf{y}|]{\alpha}$ rather than α:

$$Q_{\sqrt[|\mathbf{y}|]{\alpha}}(\mathbf{y}) = \prod_{j=1}^{|\mathbf{y}|} \sqrt[|\mathbf{y}|]{\alpha} = \alpha. \tag{3.39}$$

In initial experiments, however, we did not find any improvements in sequence length from this bias correction in practice.

Alternative loss functions that encourage smooth output distributions include explicit entropy penalization (Pereyra et al., 2017) and knowledge distillation (Sec. 3.16).

A regularization effect can also be achieved by making the training data harder to fit by adding noise, for example via subword regularization (Kudo, 2018), SwitchOut (Wang et al., 2018e), or noisy back-translation (Edunov et al., 2018a) (see Secs. 3.8.3 and 3.9).

3.11.4 Large Batch Training

Another practical trick which is becoming increasingly feasible with the availability of multi-GPU training and large GPU memories is to use very large batch sizes. Large batch training can yield almost linear speed-ups (McCandlish et al., 2018) as the computation can be distributed across multiple GPUs. Even more importantly, gradients estimated on large batches are naturally less noisy than gradients from small batches, and can yield better overall convergence (Popel and Bojar, 2018; Stahlberg et al., 2018b; Vaswani et al., 2017). For example, distributing Transformer training across 16 (effective) GPUs can improve over single GPU training by two full BLEU points (Stahlberg et al., 2018b). Smith et al. (2017) argued that increasing the batch size during training can have a similar effect as learning rate decay. For a thorough and insightful discussion of large batch training we refer the reader to (McCandlish et al., 2018).

Previous studies (Morishita et al., 2017; Neishi et al., 2017) on batch size were limited by the hardware since – in vanilla SGD – the training batch has to fit into the GPU memories. We will present a technique in Sec. 4.6 called *delayed SGD* which sidesteps these limitations by decoupling the batch size limit from the available hardware.

3.11.5 Reinforcement Learning

Ranzato et al. (2015) pointed out two weaknesses of standard MLE training in neural sequence models. First, there is a discrepancy between NMT training and decoding. During training, the correct target label y_{j-1} is used in the j-th time step. Obviously, during decoding, the correct labels are not available, so the previous (potentially wrong) output is fed back to the model. This is called 'exposure bias' (Ranzato et al., 2015) as the model is never exposed to its own mistakes during training. The exposure bias can be tackled by feeding back the ground-truth labels only at early training stages, but gradually switching to feeding back the previously produced target tokens instead as training progresses (Bengio et al., 2015).

The second issue in NMT training pointed out by Ranzato et al. (2015) is the mismatch between training loss function and evaluation metric. Training uses cross-entropy loss on the word-level, whereas the final evaluation metric is usually BLEU (Papineni et al., 2002) which is defined on sentence- or document-level. Both of these problems can be tackled with reinforcement learning (Keneshloo et al., 2018; Ranzato et al., 2015). In the standard terminology of reinforcement learning, an *agent* interacts with an *environment* via *actions*. A *policy* determines the action to pick depending on the environment. The goal is to learn a policy which maximizes the expected *reward*. In NMT, the agent is the NMT model that interacts with the environment consisting of the source sentence $\mathbf{x}$ and the translation history y_1^{j-1} by picking actions (words) according the policy $P(y_j|y_1^{j-1},\mathbf{x})$.

The advantage of casting NMT as reinforcement learning problem is that the reward does not need to be differentiable, and thus can be any quality measure such as BLEU or GLEU (Wu et al., 2016b). However, training is computationally very expensive as it requires sampling or decoding during training (Zhukov et al., 2017). Therefore, reinforcement learning is usually used to refine a model trained with cross-entropy (Wu et al., 2016b). However, even though reinforcement learning has yielded some gains in the past in isolated experiments, it is difficult to improve over stronger baselines with recent NMT architectures and back-translation (Wu et al., 2018a). Wu et al. (2016b) reported that their gains in BLEU from reinforcement learning were not reflected in the human evaluation.

Other possible applications for reinforcement learning in neural sequence prediction include architecture search (Zoph and Le, 2016), adequacy-oriented learning (Kong et al., 2018), and simultaneous translation (Sec. 3.7.8). An alternative way to incorporate the BLEU metric into NMT training is via a minimum risk formulation (Edunov et al., 2018b; Shen et al., 2016).

3.11.6 Dual Supervised Learning

Recall that NMT networks are trained to model the distribution $P(\mathbf{y}|\mathbf{x})$ over translations $\mathbf{y}$ for a given source sentence $\mathbf{x}$. This training objective takes only one translation direction into account – from the source language to the target language. However, the chain rule gives us the following relation:

$$P(\mathbf{y}|\mathbf{x})P(\mathbf{x}) = P(\mathbf{x},\mathbf{y}) = P(\mathbf{x}|\mathbf{y})P(\mathbf{y}). \tag{3.40}$$

Eq. 3.40 is often not satisfied when the two translation models $P(\mathbf{y}|\mathbf{x})$ and $P(\mathbf{x}|\mathbf{y})$ are trained independently. The dual supervised learning loss $\mathscr{L}_{\text{DSL}}$ aims to correlate both translation directions as follows (Hassan et al., 2018; Xia et al., 2017):

$$\mathscr{L}_{\text{DSL}} = \big(\log P(\mathbf{x}) + \log P(\mathbf{y}|\mathbf{x}) - \log P(\mathbf{y}) - \log P(\mathbf{x}|\mathbf{y})\big)^2. \tag{3.41}$$

An alternative way to incorporate both translation directions is the agreement-based approach of Cheng et al. (2016b).

3.11.7 Adversarial Training

Generative adversarial networks (Goodfellow et al., 2014, GANs) have recently become extremely popular in computer vision. GANs were originally proposed as framework for training generative models. For example, in computer vision, a generative model G would generate images that are similar to the ones in the training corpus. The input to a classic GAN is noise which is sampled from a noise prior. The key idea of adversarial training is that G is trained to fool a discriminative model D. The discriminator D takes an image as input and outputs the probability of the image coming from the real training corpus as opposed to being generated by G. G and D are jointly trained with opposing objectives: G tries to drive up the probability of D making a mistake whereas D aims to discriminate between real and fake images generated by G. GANs are particularly useful when they condition on some input (conditional GANs). For example, a GAN which conditions on a textual description of an image is able to synthesize an image for an unseen description at test time.

In computer vision, it is possible to back-propagate gradients through the synthetic image and thus train G and D jointly without approximations. The main challenge for applying GANs to text is that this is no longer possible since text consists of a variable number of discrete symbols. Therefore, most work on adversarial training in NLP relies on reinforcement learning to generate synthetic text samples (Li et al., 2017c; Wu et al., 2017a; Yang et al., 2018d; Yu et al., 2017; Zhang et al., 2018f) or directly operates on the hidden activations in G (Lamb et al., 2016). Besides some exploratory efforts (Wu et al., 2017a; Yang et al., 2018d; Zhang

et al., 2018f), adversarial training for NLP and particularly NMT is still in its infancy and rather brittle (Caccia et al., 2018; Michel et al., 2019; Yang et al., 2018d; Zhang et al., 2019b).

3.12 Explainable Neural Machine Translation

3.12.1 Post-hoc Interpretability

Explaining the predictions of deep neural models is hard because they consist of tens of thousands of neurons and millions of parameters. Therefore, explainable and interpretable deep learning is still an open research question (Alishahi et al., 2019; Doshi-Velez and Kim, 2017; Lipton, 2018; Montavon et al., 2018; Ribeiro et al., 2016). *Post-hoc interpretability* refers to the idea of sidestepping the model complexity by treating it as a black-box and not trying to understand the inner workings of the model. Montavon et al. (2018) defines post-hoc interpretability as follows: "A trained model is given and our goal is to understand what the model predicts (e.g. categories) in terms what is readily interpretable (e.g. the input variables)". In NMT, this means that we try to understand the target tokens ("what the model predicts") in terms of the source tokens ("the input variables"). Post-hoc intepretability methods such as layer-wise relevance propagation (Bach et al., 2015) are often visualized with heat maps representing the importance of input variables – pixels in computer vision or source words in machine translation.

Applying post-hoc interpretability methods to sequence-to-sequence prediction has received some attention in the literature (Schwarzenberg et al., 2019). Alvarez-Melis and Jaakkola (2017) proposed a causal model which finds related source-target pairs by feeding in perturbed versions of the source sentence. Ma et al. (2018c) derived relevance scores for NMT by comparing the predictive probability distributions before and after zeroing out a particular source word. See (Feng et al., 2018b) for some general limitations of such post-hoc analyses in NLP.

3.12.2 Model-intrinsic Interpretability

Unlike the black-box methods for post-hoc interpretability, another line of research tries to understand the functions of individual hidden neurons or layers in the NMT network. Different methods have been proposed to visualize the activities or gradients in hidden layers (Cashman et al., 2018; Ding et al., 2017; Karpathy et al., 2015; Li et al., 2016a). Belinkov et al. (2017) shed some light on NMT's ability to handle morphology by investigating how well a classifier can predict part-of-speech or morphological tags from the last encoder hidden layer. Bau et al. (2018); Dalvi et al. (2019, 2018) found individual neurons that capture certain linguistic

properties with different forms of regression analysis. Bau et al. (2018) were even able to alter the translation (e.g. change the gender) by manipulating the activities in these neurons. Other researchers have focused on the attention layer. Tang et al. (2018b) suggested that attention at different layers of the Transformer serves different purposes. They also showed that NMT does not use the means of attention for word sense disambiguation. Ghader and Monz (2017) provide a detailed analysis of how NMT uses attention to condition on the source sentence.

3.12.3 Confidence Estimation in Translation

Obtaining word level or sentence level confidence scores for translations is not only very useful for practical MT, it also improves the explainability and trustworthiness of the MT system. An obvious candidate for confidence scores from an NMT system are the probabilities the model assigns to tokens or sentences. However, there is some disagreement in the literature on how well NMT models are calibrated (Kumar and Sarawagi, 2019; Ott et al., 2018a). Poorly calibrated models do not assign probabilities according to the true data distribution. Such models might still assign high scores to high quality translations, but their output distributions are no reliable source for deriving word-level confidence scores. While confidence estimation has been explored for traditional SMT (Bach et al., 2011; de Gispert et al., 2013; Ueffing and Ney, 2005), it has received almost no attention since the advent of neural machine translation. The only work on confidence in NMT we are aware of is from Rikters (2018); Rikters and Fishel (2017) who aim to use attention to estimate word-level confidences.

In contrast, the related field of Quality Estimation for MT enjoys great popularity, with well-attended annual WMT evaluation campaigns – by now in their seventh edition (Specia et al., 2018). Quality estimation aims to find meaningful quality metrics which are more accepted by users and customers than abstract metrics like BLEU (Papineni et al., 2002), and are more correlated to the usefulness of MT in a real-world scenario. Possible applications for quality estimation include estimating post-editing efficiency (Specia, 2011) or selecting sentences in the MT output which need human revision (Bach et al., 2011).

3.12.4 Word Alignment in Neural Machine Translation

Word alignment such as discussed in Sec. 2.4 is one of the fundamental problems in traditional phrase-based SMT. SMT constructs the target sentence by matching phrases in the source sentence, and combing their translations to form a fluent sentence (Chiang, 2007; Koehn, 2010). This approach does not only yield a translation, it also produces a word alignment along with it since each target phrase is generated from a unique source phrase. Thus, a word alignment can be seen as an explanation for the produced translation: each target phrase is explained with a

link into the source sentence. Vanilla NMT does not have the notion of a hard word alignment, and we will demonstrate in Sec. 8.4 that attention is a poor substitute for it as it serves a very different purpose.

Despite considerable consensus about the importance of word alignments for practical machine translation (Koehn and Knowles, 2017), e.g. to enforce constraints on the output (Hasler et al., 2018) or to preserve text formatting, introducing explicit alignment information to NMT is still an open research problem. Word alignments have been used as supervision signal for the NMT attention model (Alkhouli and Ney, 2017; Chen et al., 2016; Liu et al., 2016b; Mi et al., 2016b). Cohn et al. (2016) showed how to reintroduce concepts known from traditional statistical alignment models (Brown et al., 1993) like fertility and agreement over translation direction to NMT.

Hard attention (Xu et al., 2015) is a discrete version of the usual soft attention and is thus closer to the concept of a hard alignment. Similar ideas have been explored for speech recognition (Lawson et al., 2018), morphological inflection (Aharoni and Goldberg, 2017a), text summarization (Raffel et al., 2017; Yu et al., 2016b), and image caption generation (Xu et al., 2015). Some approaches to simultaneous translation presented in Sec. 3.7.8 explicitly control for reading source tokens and writing target tokens and thereby generate monotonic hard alignments on the segment level (Gu et al., 2017c; Yu et al., 2016a). Hybrids between soft and hard attention have been proposed by Choi et al. (2017a); Shen et al. (2018c). However, the usefulness of hard attention for generic offline machine translation is often limited since it usually can only represent monotonic alignments.

Alkhouli et al. (2016) used separate alignment and lexical models and thus were able to hypothesize explicit alignment links during decoding. Alignment-based NMT has been extended to multi-head attention by using an additional alignment head (Alkhouli et al., 2018). A similar idea was pursued by Zenkel et al. (2019) who added an additional alignment layer to the Transformer and trained it – unlike Alkhouli et al. (2018) – in an unsupervised way.

3.13 Alternative NMT Architectures

3.13.1 Extensions to the Transformer Architecture

The Transformer model architecture (Vaswani et al., 2017) introduced in Sec. 3.6.5 has become the de facto standard architecture for neural machine translation because of its superior translation quality on a variety of language pairs (Bojar et al., 2019, 2018).[15] The Transformer

[15] The only contrary evidence we are aware of is from Tran et al. (2018) who found that recurrent models can better model hierarchical structure than the Transformer.

comes with a number of techniques which sets it apart from previous architectures such as multi-head attention, self-attention, large batch training, etc. Some ablation studies in the literature aim to factor out or explain the contributions of these different techniques (Chen et al., 2018b; Domhan, 2018; Tang et al., 2018a,b). Several attempts have been made to improve different aspects of the vanilla model for machine translation, but none has been widely adopted. Most notably, Shaw et al. (2018) proposed to embed relative positions rather than absolute ones. We will confirm the potential of relative positioning in our own experiments in Sec. 4.6. A disadvantage of the relative Transformer is the increased computational complexity. The memory keys and values with absolute positions are the same in each decoding step. With relative positioning, however, both have to be recomputed in each time step since the relative positions change over time. The model of Song et al. (2018) works with attention masks (Sec. 3.6.2) to narrow down context. Ahmed et al. (2017) proposed to weight the output of attention heads inside multi-head attention. The Star-Transformer (Guo et al., 2019) thins out inter-layer connections of the standard model to reduce computational complexity. With a similar outset, Medina and Kalita (2018) reported speed-ups by replacing the single deep encoder with multiple shallow encoders.

Some recent research has focused on large scale language modelling with the Transformer (Dai et al., 2019; Krause et al., 2019; Radford et al., 2018, 2019; Wang et al., 2019a). This line of research seems to be particularly prone to marketing hypes, with OpenAI famously raising some eyebrows by proclaiming that due to "concerns about malicious applications of the technology, we are not releasing the trained model".[16] The Transformer is also the starting point for neural architectures for contextualized word embeddings (see Sec. 3.2) such as BERT (Devlin et al., 2019).

3.13.2 Advanced Attention Models

As shown in Sec. 3.6.1, the vast majority of current NMT architectures are based on one of three attention types: additive, (scaled) dot-product, or multi-head attention (Bahdanau et al., 2015; Luong et al., 2015b; Vaswani et al., 2017). In this section, we will outline attempts to improve upon these standard models.

Sec. 3.10.1 discussed the problem of over- and under-translation, and how coverage models can mitigate this problem by controlling the attention weights with fertilities. Alternatively, researchers have tried to equip the attention layer itself with additional components like a memory (Meng et al., 2018) or a recurrent network (Feng et al., 2016; Yang et al., 2017b) to enable it to keep track of the attention history. Choi et al. (2018b) proposed an attention model

[16] https://openai.com/blog/better-language-models/

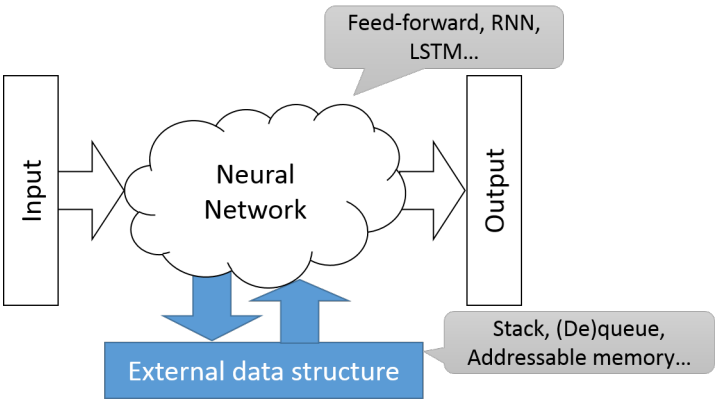


Fig. 3.23 Neural networks with external memory.

that is able to learn different attention weights for each dimension in the values, not only one weight for each value vector.

One potential weakness of the standard models is that they are token-based: the attention output is a weighted average of the values, and the attention weights tend to focus on a single key-value pair. Therefore, there is no explicit mechanism to attend to full phrases rather than subwords or characters.[17] Phrase-based NMT which equips the model with the ability to attend to full phrases or multi-word expressions has been studied by Eriguchi et al. (2019); Feng et al. (2018a); Huang et al. (2017b); Ishiwatari et al. (2017); Li et al. (2017e); Rikters and Bojar (2017).

On the other side of the spectrum, it has been noted that regular attention sometimes spreads out over too many elements, especially when applied over long sequences. The attention output in this case is an average of many values which is naturally more noisy than with sharp attention, and which impedes the propagation of information through the network. Hard attention (Sec. 3.12.4) removes this sort of noise, but is often restricted to monotonic alignment. Lin et al. (2018a) proposed to explicitly learn to set the temperature of attention weights to control the softness of attention. Another potential solution has been suggested by Zhang et al. (2017a) who used GRU gates rather than weighted linear combinations to compute the attention output from the values.

3.13.3 Memory-augmented Neural Networks

RNNs are theoretically Turing-complete (Siegelmann and Sontag, 1995) and thus potentially very powerful models of computation. However, since training is still a challenge (see Sec. 3.11),

[17] This does not mean that the source sentence context is always reduced to a single input token since the encoder hidden states are by themselves context-sensitive.

even advanced RNN architectures like LSTMs (Hochreiter and Schmidhuber, 1997) fail to solve certain basic sequence-to-sequence tasks like (repeated) copying or reversal in practice (Graves et al., 2014; Grefenstette et al., 2015). This observation motivated researchers to add external memory structures like a memory tape (Williams and Zipser, 1989) or a stack (Sun et al., 1990, 1993) to the neural network. The basic idea is illustrated in Fig. 3.23. Besides producing the output sequence, the neural network learns to operate an external data structure. The external memory is not part of the neural network but the network learns to communicate with it through conceptually discrete operations like PUSH and POP. However, in order to train the whole system with a gradient-based optimizer, these discrete operations are often approximated with continuous versions (Graves et al., 2014; Grefenstette et al., 2015; Joulin and Mikolov, 2015). Various data structures have been used in combination with neural networks such as (inter alia) stacks (Joulin and Mikolov, 2015), (double-ended) queues (Grefenstette et al., 2015), addressable memory cells (Graves et al., 2014, 2016; Kurach et al., 2015), and hierarchical memory structures (Chandar et al., 2016). Grefenstette et al. (2015) suggested that even simple data structures like dequeues help to solve linguistically motivated tasks like bigram flipping or Inversion Transduction Grammar (Wu, 1997, ITG) tasks.

Research on these kinds of neural network operated data structures still mainly focuses on synthetic tasks like relatively simple algorithmic problems. Initial efforts to apply this line of research to real world problems are limited to neural machine translation (Feng et al., 2017; Li et al., 2019; Wang et al., 2016; Xiong et al., 2018a), sentence simplification (Vu et al., 2018), and text normalization (Pramanik and Hussain, 2019).

3.13.4 Beyond Encoder-decoder Networks

All NMT architectures which we have discussed in the previous sections fall in the category of encoder-decoder networks: An encoder network computes a fixed or variable length continuous hidden representation of the source sentence, and a separate decoder network defines a probability distribution over target sentences given that representation. There are some initial efforts in the literature to depart from this overall structure. For example, variational methods that define a *distribution* over (a part of) the hidden representations have been explored by Bastings et al. (2019); Shah and Barber (2018); Su et al. (2018); Zhang et al. (2016). Non-autoregressive NMT which aims to reduce or remove the sequential dependency on the translation prefix inside the decoder for enhanced parallelizability has been studied by Akoury et al. (2019); Gu et al. (2017a); Guo et al. (2018); Lee et al. (2018); Libovický and Helcl (2018); Wang et al. (2018a, 2019b). Bahar et al. (2018); Kaiser and Bengio (2016) recomputed the encoder state after each time step and thus effectively expanded the hidden representation into a 2D structure. The architecture proposed by He et al. (2018a) does not only use the last encoder layer as

hidden representation, but instead connects encoder and decoder layers at the same depth via attention.

3.14 Data Sparsity

Deep learning methods are notoriously data hungry. For example, traditional statistical machine translation still often outperforms neural machine translation when training data is scarce (Koehn and Knowles, 2017; Östling and Tiedemann, 2017). In this section we will look at the problem of training data sparsity from different angles such as reducing noise in training data (Sec. 3.14.1), using data from a different domain (Sec. 3.14.2), or making use of less or no parallel data (Secs. 3.14.3-3.14.4). In Sec. 7.4 we will present our language model based approach which is especially suitable for low-resource NMT.

3.14.1 Corpus Filtering

Unfortunately, MT training data is usually inherently noisy as it is often extracted (semi-) automatically by crawling the web (Resnik, 1999; Resnik and Smith, 2003) and therefore commonly contains sentence fragments, wrong languages, misaligned sentence pairs (Khayrallah and Koehn, 2018), or MT output rather than genuine parallel text (Arase and Zhou, 2013; Rarrick et al., 2011). In the previous sections we discussed several instances of the use of synthetic noise in NMT. For example, adding noise to the synthetic sentences in back-translation can be beneficial (Sec. 3.9). Noise can also be used to generate diverse translations (Sec. 3.7.7) or as regularizer (Sec. 3.11.3). However, when discussing the role of noise in NMT it is imperative to carefully differentiate between the various kinds of noise and the ways it impacts NMT. Studies have shown that NMT is not robust against naturally occurring noise at training (Khayrallah and Koehn, 2018) and test (Belinkov and Bisk, 2017; Cheng et al., 2018; Michel and Neubig, 2018; Ruiz et al., 2017) time. Robustness at test time can be improved by training on synthetic noise (Karpukhin et al., 2019; Vaibhav et al., 2019). Corpus filtering to reduce the amount of noise in the training data has been widely studied for traditional SMT (Cui et al., 2013; Taghipour et al., 2011), often in context of domain adaptation (Axelrod et al., 2011; Foster et al., 2010). More recent research on data filtering focuses on NMT since van der Wees et al. (2017) had shown that filtering techniques developed for SMT are less useful for NMT. One of the first approaches to NMT corpus filtering was the method of Carpuat et al. (2017) based on semantic analysis. The most effective approaches in the WMT18 shared task on corpus filtering for NMT (Koehn et al., 2018) used a combination of likelihood scores from neural translation models and neural language models which have been trained on clean data (Junczys-

Dowmunt, 2018a,b; Rossenbach et al., 2018). These criteria prefer sentence pairs which are likely translations of one another according the translation model (Xu et al., 2018). Zhang et al. (2017b) proposed the exact opposite, arguing that NMT training should concentrate on "difficult" training samples, i.e. samples with low translation probability. An alternative to hard data filtering called *curriculum learning* (Bengio et al., 2009) that controls the order of training samples has been applied to NMT by Kumar et al. (2019); Platanios et al. (2019); van der Wees et al. (2017); Wang et al. (2018d).

3.14.2 Domain Adaptation

There is a robust body of research on domain adaptation for machine translation (Chu et al., 2018; Chu and Wang, 2018). Popular domain adaptation techniques for both SMT and NMT aim to select (Axelrod et al., 2011; Foster et al., 2010; Hildebrand et al., 2005; Wang et al., 2017b, 2018c) or weight (Chen et al., 2017a; Wang et al., 2018c, 2017c) samples in a large out-of-domain corpus. Back-translation (Sec. 3.9) can also be used for domain adaptation by back-translating sentences from an in-domain monolingual corpus. Another simple yet very effective method is to jointly train on in-domain and out-domain sentences, possibly with domain-tags to help learning (Britz et al., 2017; Kobus et al., 2017; Tars and Fishel, 2018). Sajjad et al. (2017) showed that a simple concatenation of in-domain and out-domain corpora can already increase the robustness and generalization of NMT significantly. Khayrallah et al. (2017) studied domain adaptation by constraining an NMT system to SMT lattices. Their system is very similar to our work on syntactically guided NMT in Sec. 4.3 although we did not target domain adaptation. Freitag and Al-Onaizan (2016) ensembled separately trained general-domain and in-domain models.

Another widely used technique is to train the model on a general domain corpus, and then fine-tune it by continuing training on the in-domain corpus (Luong and Manning, 2015; Sennrich et al., 2016b). Fine-tuning bears the risk of two negative effects: catastrophic forgetting (French, 1999; Goodfellow et al., 2013a) and over-fitting. Catastrophic forgetting occurs when the performance on the specific domain is improved after fine-tuning, but the performance of the model on the general domain has decreased drastically. The risk of over-fitting is connected to the fact that the in-domain corpus is usually very small. Both effects can be mitigated by artificially limiting the learning capabilities of the fine-tuning stage, e.g. by freezing sub-networks (Thompson et al., 2018) or by only learning additional scaling factors for hidden units rather than full weights (Swietojanski and Renals, 2014; Vilar, 2018). A very elegant way to prevent over-fitting and catastrophic forgetting is to apply regularizers (Sec. 3.11.3) to keep the adapted model weights close to their original values. Dakwale and Monz (2017); Khayrallah et al. (2018) regularized the output distributions using

techniques inspired by knowledge distillation (Sec. 3.16). Miceli Barone et al. (2017) applied standard L2 regularization and a variant of dropout to domain adaptation. Elastic weight consolidation (Kirkpatrick et al., 2017) can be seen as generalization of L2 regularization that takes the importance of weights (in terms of Fisher information) into account, and has been applied to NMT domain adaptation by Saunders et al. (2019); Thompson et al. (2019). In particular, Saunders et al. (2019) showed that EWC does not only reduce catastrophic forgetting but even yields gains on the general domain when used for fine-tuning on a related domain.

3.14.3 Low-resource NMT

One of the areas in which traditional SMT still often outperforms NMT is low-resource translation (Koehn and Knowles, 2017; Östling and Tiedemann, 2017). However, several techniques have been proposed to improve the performance of NMT under low-resource conditions. In general, the methods discussed in Sec. 3.9 to leverage monolingual data such as back-translation are particularly effective for low-resource MT. Ren et al. (2018) proposed a scheme that could make use of translations from/into the source/target language into/from a third resource-rich language. The transfer-learning approach of Zoph et al. (2016) first trains a *parent* model on a resource-rich language pair (e.g. French-English), and then continues training on the low-resource pair of interest (e.g. Uzbek-English). The effectiveness of transfer-learning depends on the relatedness of the languages (Dabre et al., 2017; Murthy et al., 2019; Nguyen and Chiang, 2017; Zoph et al., 2016). The rapid adaptation of multilingual NMT systems to new low-resource language pairs has been studied by Neubig and Hu (2018). Approaches that do not rely on resources from a third language include Östling and Tiedemann (2017) who supervised the generation order of an insertion-based low-resource translation model with word alignments.

A series of NIST evaluation campaigns called LoReHLT (Tong et al., 2018) focuses on low-resource MT, and recent WMT editions also contain low-resource language pairs (Bojar et al., 2017, 2019, 2018).

3.14.4 Unsupervised NMT

Unsupervised NMT is an extreme case of the low-resource scenario in which not even small amounts of cross-lingual data is available, and the translation system learns entirely from (unrelated) monolingual data. Unsupervised NMT often starts off from an unsupervised cross-lingual word embedding model (Artetxe et al., 2017a; Conneau et al., 2017b; Hoshen and Wolf, 2018) that maps word embeddings from the source and the target language into a joint embedding space (Artetxe et al., 2017b; Lample et al., 2017). The translation model is then

further refined by iterative back-translation (Lample et al., 2018; Ren et al., 2019). The extract-edit scheme of Wu et al. (2019b) is an alternative to back-translation for unsupervised NMT that edits a sentence in the monolingual corpus rather than synthesize it from scratch. Unsupervised NMT has been targeted in recent WMT evaluation campaigns (Bojar et al., 2019, 2018).

3.15 Multilingual NMT

NMT is usually trained to translate a single fixed source language into another fixed target language. Multilingual NMT aims to cover translation directions between multiple languages with a single model. This does not only have the potential of exploiting similarities across language pairs, it also reduces the number of systems required for all-way translation between a set of languages from quadratic to linear or even one. Multilingual NMT systems can be largely categorized by the components they share between language directions. On one side of the spectrum, the entire neural architecture (both encoder and decoder) can be shared, and source and target languages can be specified by annotating sentences (Johnson et al., 2017) or words (Ha et al., 2016, 2017) with language ID tags or embeddings. On the other side of the spectrum, Luong et al. (2015a) used a separate encoder for each source language and a separate decoder for each target language. Firat et al. (2016a, 2017) extended the work of Luong et al. (2015a) to attentional NMT by sharing the attention mechanism across language directions. Dong et al. (2015) studied one-to-many translation with a single encoder but separate decoders for each target language.

A potential benefit of multilingual systems is zero-shot translation, i.e. the translation between two languages for which no direct training data is available.[18] Johnson et al. (2017) reported reasonable Portuguese→Spanish translation performance of their multilingual system that has been trained on Portuguese↔English and Spanish↔English, although pivoting through English (translate Spanish to English, and then English to Portuguese) worked better. Pivot-based zero-shot translation can be further improved by fine-tuning on a pseudo parallel corpus (Firat et al., 2016b) or by jointly training some components of the source-pivot and pivot-target systems like word embedding matrices (Cheng et al., 2017). Lu et al. (2018) reported gains in zero-shot settings by adding a boldly named "neural interlingual" component between the encoder and the decoder which is shared across language directions. For an assessment of the current capabilities of multilingual and zero-shot translation systems see (Aharoni et al., 2019; Cettolo et al., 2017; Lakew et al., 2018).

[18] The difference between zero-shot and unsupervised NMT (Sec. 3.14.4) is that unsupervised NMT does not rely on *any* cross-lingual data whereas zero-shot NMT uses cross-lingual data in other language directions.

Another form of multilingual NMT is multi-source NMT (Och and Ney, 2001; Zoph and Knight, 2016), in which the system tries to generate a single translation given sentences in two source languages simultaneously. A problem with this approach is data sparsity as missing source sentences have to be synthesized (Choi et al., 2018a; Nishimura et al., 2018) if the training corpus does not provide sentences in all source languages. In a wider context, multi-source architectures can be used for multimodal NMT (Sec. 3.17.1), morphological inflection (Kann et al., 2017), zero-shot translation (Firat et al., 2016b), low-resource MT (Choi et al., 2018a), syntax-based NMT (Currey and Heafield, 2018), document-level MT (Bawden et al., 2018), or bidirectional decoding (Li et al., 2017a).

Dabre et al. (2019) provide a comprehensive overview of recent trends in multilingual NMT.

3.16 NMT Model Size

NMT models usually have hundreds of millions of parameters (Tab. 3.3). Such large models cause a number of practical issues. GPUs are usually required to run such big models efficiently, but GPUs are expensive and their memory is limited. Smaller models would not only reduce the computational complexity but could also make better use of GPU parallelism by increasing batch sizes. Furthermore, model files require large amounts of disk space which is a problem on mobile platforms.

One way to increase the space efficiency of neural models is neural architecture search (Wang et al., 2019a; Zoph and Le, 2016). For example, So et al. (2019) found computationally efficient Transformer hyper-parameters by systematic neural architecture search. Rather than optimizing the dimensionality of layers, it is also possible to significantly speed up translation by departing from the usual 32 bit floating point arithmetics by reducing the precision to 8 or 16 bits (Devlin, 2017; Hoang et al., 2018a; Ott et al., 2018b; Quinn and Ballesteros, 2018) or by using vector quantization (Wu et al., 2016a,b).

The approach we will pursue in Ch. 6 is to train a large neural network and shrink it afterwards. The idea of pruning neural networks to improve the compactness of the models dates back almost 30 years (LeCun et al., 1989b). The literature is therefore vast (Augasta and Kathirvalavakumar, 2013). One line of research aims to remove unimportant network connections. The connections can be selected for deletion based on the second-derivative of the training error with respect to the weight (Hassibi et al., 1993; LeCun et al., 1989b), or by a threshold criterion on its magnitude (Han et al., 2015). See et al. (2016) confirmed a high degree of weight redundancy in NMT networks. Zhu and Gupta (2017) demonstrated that large sparse models outperform smaller dense networks with the same memory footprint.

In Ch. 6, we will argue that removing neurons rather than individual connections does not only improve the model size but also the decoding speed. Srinivas and Babu (2015) proposed to remove neurons which are very similar to another neuron and have small outgoing weights. Babaeizadeh et al. (2016) combined pairs of neurons with similar activities during training.

Using low rank matrices for neural network compression, particularly approximations via Singular Value Decomposition (SVD), has been studied widely in the literature (Denil et al., 2013; Denton et al., 2014; Lu et al., 2016; Prabhavalkar et al., 2016; Xue et al., 2013).

Another approach, known as *knowledge distillation*, uses a large model (the teacher) to generate soft training labels for a smaller student network (Buciluǎ et al., 2006; Hinton et al., 2015). The student network is trained by minimizing the cross-entropy to the teacher. This idea has been applied to sequence modelling tasks such as machine translation and speech recognition (Freitag et al., 2017; Kim et al., 2019a; Kim and Rush, 2016; Liu et al., 2019; Wong and Gales, 2016; Zhang et al., 2018a).

3.17 NMT with Extended Context

3.17.1 Multimodal NMT

Machine translation is usually framed as the isolated transformation of the textual representation of a single sentence in one language into another. Since language is inherently ambiguous, researchers have searched for ways to provide the translation system with more context. For example, if the source sentence describes an image, the image itself potentially carries valuable clues to help the translation process. Multimodal machine translation (Elliott et al., 2015; Hitschler et al., 2016) aims to generate an image caption in the target language given both the source language caption and the image itself. The core of most multimodal MT models is a normal text-to-text system which integrates visual information by using global image features extracted with a separate computer vision model (Elliott et al., 2015; Hitschler et al., 2016) or via visual attention (Huang et al., 2016). Multimodality in translation was the subject of a series of WMT shared tasks (Barrault et al., 2018; Elliott et al., 2017; Specia et al., 2016). Calixto and Liu (2019) demonstrated the usefulness of visual clues in translation.

3.17.2 Tree-based NMT

The prevalent choice for modeling units in NMT are characters are subword-units (Sec. 3.8.3). This design decision is not linguistically motivated but rather stems from the difficulty of extending NMT to an open vocabulary. From the linguistic perspective, however, translation is

better viewed as the transformation of larger elements in the sentence such as words, phrases, or even syntactic structures.

Various attempts have been made to introduce structures such as syntactic constituency trees or dependency trees both on the source and the target side of NMT. A popular approach is to retain the sequence-to-sequence architecture and linearize the tree structures, for example using bracket expressions (Aharoni and Goldberg, 2017b; Currey and Heafield, 2018; Ma et al., 2017; Vinyals et al., 2015), sequences of rules (Saunders et al., 2018), or CCG supertags (Nadejde et al., 2017). Ma et al. (2018a); Zaremoodi and Haffari (2018) developed a linearization of a packed forests that represented multiple source sentence parses. Saunders et al. (2018) reported gains by ensembling different linearization strategies of target-side syntax trees. We will take a closer look at multi-representation ensembles in Sec. 8.2. Recurrent neural network grammars (Dyer et al., 2016) that represent syntactic parse trees as sequence of actions were applied to machine translation by Bradbury and Socher (2017); Eriguchi et al. (2017). Using actions to build target side tree structures is also central to the tree-based decoders of Wang et al. (2018f); Wu et al. (2017b). Akoury et al. (2019) used syntax to speed up decoding by first predicting a parse tree, and then predicting all target tokens in parallel.

Tree-LSTMs (Tai et al., 2015) make it possible to represent a tree structure directly with the neural network architecture. They are a generalization of recurrent LSTM cells (Sec. 3.6.3) that replaces the single input of a standard LSTM cell (usually from the previous time step) with multiple input connections, one from each child node. Thus, each Tree-LSTM cell represents a node in the tree, and the root node contains a fixed-length vector encoding of the whole tree structure. Tree-LSTMs have been applied to syntax-based NMT (Chen et al., 2017b; Eriguchi et al., 2016; Yang et al., 2017a). An alternative to Tree-LSTMs was proposed by Shen et al. (2019) who rearranged neurons in an LSTM network to resemble a block representation of the tree. Bastings et al. (2017); Chen et al. (2017c) used convolutional encoders to represent a dependency graph in the source sentence. Chen et al. (2018a) biased encoder-decoder attention weights with syntactic clues. Unsupervised tree-based methods have been studied by Kim et al. (2019b); Maillard et al. (2017); Williams et al. (2018).

3.17.3 NMT with Graph Structures as Input

As a generalization of the tree-based approaches discussed in the previous section, lattice-based NMT allows more general graph structures on the input side to provide a richer description of the source sentence. Lattices can represent uncertainty of upstream components such as speech recognizers (Sperber et al., 2017) or tokenizers (Su et al., 2017; Tan et al., 2018). Lattices have also been used to augment the input with external knowledge sources such

as knowledge graphs (Koncel-Kedziorski et al., 2019; Moussallem et al., 2019) or semantic predicate-argument structures (Marcheggiani et al., 2018).

Factors are another way of providing more information to the translation system. Factors describe a word by a tuple consisting of its lemma and various linguistic information (prefix, suffix, part-of-speech etc.) rather than its surface form. This technique is popular for traditional statistical machine translation (Koehn, 2010; Koehn and Hoang, 2007), and has been applied to neural machine translation both on the input (Sennrich and Haddow, 2016) and the output (García-Martínez et al., 2016; García-Martínez et al., 2017) side.

3.17.4 Document-level Translation

MT systems usually translate sentences in isolation. However, there is evidence that humans also take context into account, and rate translations from humans with access to the full document higher than the output of a state-of-the-art sentence-level machine translation system (Läubli et al., 2018). Common examples of ambiguity which can be resolved with cross-sentence context are pronoun prediction or coherency in lexical choice.

Various techniques have been proposed to provide the translation system with inter-sentential context, for example by initializing encoder or decoder states (Wang et al., 2017a), using multi-source encoders (Bawden et al., 2018; Jean et al., 2017), as additional decoder input (Wang et al., 2017a), with memory-augmented neural networks (Kuang et al., 2017; Maruf and Haffari, 2018; Tu et al., 2018), hierarchical attention (Maruf et al., 2019; Miculicich et al., 2018b), deliberation networks (Xiong et al., 2018b), or by simply concatenating multiple source and/or target sentences (Bawden et al., 2018; Tiedemann and Scherrer, 2017). Context-aware extensions to Transformer encoders have been proposed by Voita et al. (2018); Zhang et al. (2018b). Techniques also differ in whether they use source context only (Jean et al., 2017; Voita et al., 2018; Wang et al., 2017a; Zhang et al., 2018b), target context only (Kuang et al., 2017; Tu et al., 2018), or both (Bawden et al., 2018; Maruf and Haffari, 2018; Maruf et al., 2019; Miculicich et al., 2018b; Tiedemann and Scherrer, 2017). Several studies on document-level NMT indicate that automatic and human sentence-level evaluation metrics often do not correlate well with improvements in discourse level phenomena (Bawden et al., 2018; Läubli et al., 2018; Müller et al., 2018).

3.18 Conclusion

Neural machine translation (NMT) has become the de facto standard for large-scale machine translation in a very short period of time. This chapter traced back the origin of NMT to word

and sentence embeddings and neural language models. We reviewed the most commonly used building blocks of NMT architectures – recurrence, convolution, and attention – and discussed popular concrete architectures such as RNNsearch, GNMT, ConvS2S, and the Transformer. We discussed the advantages and disadvantages of several important design choices that have to be made to design a good NMT system with respect to decoding, training, and segmentation. We then explored advanced topics in NMT research such as explainability and data sparsity. Most of these topics have clear links to our own work presented in later chapters:

- Ch. 5 is devoted to NMT decoding discussed in Sec. 3.7.
- In Ch. 7 we present various ways to use language models for NMT which relates to Sec. 3.9.
- NMT model errors and search errors (Sec. 3.10) are studied thoroughly in Sec. 5.4.4.
- Explainability (Sec. 3.12) is our prime motivation for our work on operation sequence NMT in Sec. 8.4.
- Our NMT network shrinking method in Ch. 6 takes up Sec. 3.16 on NMT model sizes.
- We follow up on document-level NMT (Sec 3.17.4) in Sec. 7.5 with a document-level Transformer language model.

Other sections are just included for the sake of completeness to provide a comprehensive picture of recent trends in NMT research such as the alternative architectures in Sec. 3.13, unsupervised NMT in Sec. 3.14.4, multilingual NMT in Sec. 3.15, or multimodal NMT in Sec. 3.17.1.

This chapter concludes the literature review part. The next chapters are devoted to the original contributions of this thesis, starting with SMT-NMT hybrid systems that combine statistical machine translation (Ch. 2) and neural machine translation (this chapter) approaches.

The more I think about language, the more it amazes me that people ever understand each other at all.

Kurt Gödel

4 SMT-NMT Hybrid Systems

This chapter draws from the following publications: Stahlberg et al. (2019b) in Sec. 4.2, Stahlberg et al. (2016b) in Sec. 4.3, Stahlberg et al. (2016a) in Sec. 4.4, Stahlberg et al. (2017a) in Sec. 4.5, Stahlberg et al. (2018b) in Sec. 4.6. The material has been extended, restructured, and linked with other parts of this thesis, but some passages have been quoted verbatim from these sources.

4.1 Motivation

We showed in Ch. 3 how neural models were increasingly used as features in traditional SMT until neural machine translation (NMT) evolved as new paradigm. Without question, NMT has become the prevalent approach to machine translation in recent years. However, we believe that the recent success of NMT does not necessarily have to imply a complete departure from traditional MT, and that combining both paradigms carries great potential. Neubig et al. (2015) and Stahlberg et al. (2016b) suggest that syntactic machine translation such as Hiero (Chiang, 2007) and NMT are very different and have complementary strengths and weaknesses. In particular, we emphasize the following distinctive features of Hiero and NMT.

- Hiero and other symbolic systems are easily trained with large vocabularies as required for open-domain MT, but extending pure NMT to large vocabularies is still an open research question (see Sec. 3.8).
- NMT has the advantage of including long-range context in modelling individual translation hypotheses. Hiero considers a much bigger search space, and has an n-gram language model, but a much simpler translation model.
- Standard NMT does not have an explicit mechanism for tracking source coverage, and there is evidence that may lead to both 'over-translation' and 'under-translation' (cf. Sec. 3.10.1). By contrast, Hiero source language parses completely cover the source sentence (Chiang et al., 2009).

Therefore, in this chapter we present different methods to combine the strengths of NMT and SMT. We propose three different combination schemes. The first one (Sec. 4.3) is called 'syntactically guided neural machine translation' (Stahlberg et al., 2016b) and is related to translation lattice rescoring. This approach forces the NMT decoder to select a translation from a Hiero lattice. The second approach (Sec. 4.4) allows NMT to diverge from Hiero as it combines both systems via an edit distance based similarity measure (Stahlberg et al., 2016a). The third approach (Sec. 4.5) also allows a loose combination of NMT and SMT as it biases an unconstrained NMT decoder towards n-grams which are likely according the SMT system (Stahlberg et al., 2017a). We have demonstrated the effectiveness of our hybrid schemes under evaluation conditions with three successful submissions to WMT competitions (in the years 2016, 2018, and 2019) for English-German, German-English, and Chinese-English (Stahlberg et al., 2018b, 2016a, 2019b).

This chapter contains the technical descriptions of our approaches and results. Ch. 5 will discuss our implementation of these schemes inside our decoding framework SGNMT.

4.2 Related Work

There is a large body of research comparing NMT and SMT (Tab. 4.1). Most studies have found superior overall translation quality of NMT models in most settings, but complementary strengths of both paradigms. Therefore, the literature about hybrid NMT-SMT systems is also vast. We distinguish between two categories of approaches for blending SMT and NMT.

Approaches in the first category do not employ a full SMT system but borrow only key ideas or components from SMT to address specific issues in NMT. It is straight-forward to combine NMT scores with other features normally used in SMT (like language models) in a

Neural machine translation	Statistical machine translation
+ Much better overall translation quality than SMT with enough training data (Bentivogli et al., 2016, 2018; Castilho et al., 2017b; Junczys-Dowmunt et al., 2016a; Koehn and Knowles, 2017; Toral and Sánchez-Cartagena, 2017; Volkart et al., 2018).	+ Outperforms NMT in low-resource scenarios (Dowling et al., 2018; Jauregi Unanue et al., 2018; Koehn and Knowles, 2017; Mahata et al., 2018; Menacer et al., 2017; Ojha et al., 2018).
+ More fluent than SMT (Bentivogli et al., 2016; Castilho et al., 2017a,b; Mahata et al., 2018; Toral and Sánchez-Cartagena, 2017).	+ Produces richer output lattices (Stahlberg et al., 2016b).
+ Better handles a variety of linguistic phenomena than SMT (Bentivogli et al., 2016, 2018; Isabelle et al., 2017).	+ More robust against noise (Khayrallah and Koehn, 2018; Ruiz et al., 2017).
− Adequacy issues due to lack of explicit coverage mechanism (Castilho et al., 2017a; Kong et al., 2018; Mahata et al., 2018; Tu et al., 2016; Yang et al., 2018a).	+ Translation quality degrades less on very long sentences than NMT (Bentivogli et al., 2016; Toral and Sánchez-Cartagena, 2017).
− Lack of hypothesis diversity (Sec. 3.7.7).	+ Less errors in the translation of proper nouns (Bentivogli et al., 2018).
− Neural models perform not as well as specialized symbolic models on several monotone seq2seq tasks (Schnober et al., 2016).	○ NMT and SMT require comparable amounts of (document-level) post-editing (Castilho et al., 2017b; Jia et al., 2019a).

Table 4.1 Summary of studies comparing traditional statistical machine translation and neural machine translation.

log-linear model (Gulcehre et al., 2015; He et al., 2016d)[1]. Conventional symbolic SMT-style lexical translation tables can be incorporated into the NMT decoder by using the soft alignment

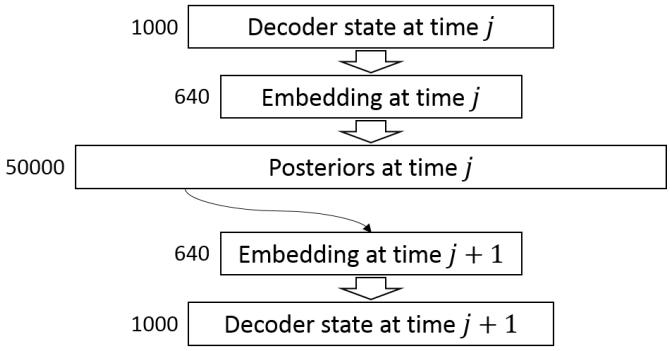
[1]Note that this is still different from using neural features in an SMT system as the standard left-to-right NMT decoder is used.

weights of the standard NMT attention model (Arthur et al., 2016; He et al., 2016d; Neubig, 2016; Tang et al., 2016; Zhang and Zong, 2016a). Cohn et al. (2016) proposed to enhance the attention model in NMT by implementing basic concepts from the original word alignment models (Brown et al., 1993; Vogel et al., 1996) like fertility and relative distortion.

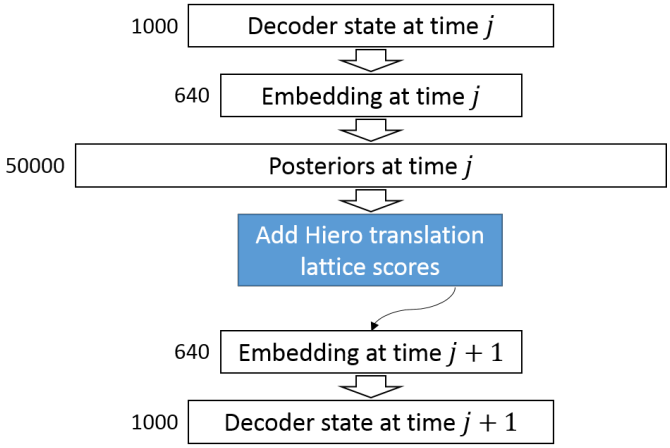
The second category of hybrid systems is related to system combination, which includes all our methods proposed in the later sections of this chapter. The idea is to combine a fully trained SMT system with an independently trained NMT system. Popular examples in this category are rescoring and reranking methods (Avramidis et al., 2016; Grundkiewicz and Junczys-Dowmunt, 2018; Khayrallah et al., 2017; Marie and Fujita, 2018; Neubig et al., 2015; Stahlberg et al., 2016b; Zhang et al., 2017d), although these models may be too constraining if the neural system is much stronger. We will discuss our loose combination schemes in Secs. 4.4 and 4.5. NMT and SMT can also be combined in a cascade, with SMT providing the input to a post-processing NMT system (Niehues et al., 2016; Zhou et al., 2017) or vice versa (Du and Way, 2017). Wang et al. (2017d, 2018g) interpolated NMT posteriors with word recommendations from SMT and jointly trained NMT together with a gating function which assigns the weight between SMT and NMT scores dynamically. The AMU-UEDIN submission to WMT16 let SMT take the lead and used NMT as a feature in phrase-based MT (Junczys-Dowmunt et al., 2016b). In contrast, Long et al. (2016) translated most of the sentence with an NMT system, and just used SMT to translate technical terms in a post-processing step. Dahlmann et al. (2017) proposed a hybrid search algorithm in which the neural decoder expands hypotheses with phrases from an SMT system. SMT can also be used as regularizer in unsupervised NMT (Ren et al., 2019).

4.3 Syntactically Guided Neural Machine Translation

Decoding in NMT is usually based on simple beam search with a narrow beam (Sec. 3.7). Syntactically guided neural machine translation (Stahlberg et al., 2016b) is motivated by the idea that the structured search spaces defined by syntactic machine translation approaches such as Hiero (Chiang, 2007) can be used to guide NMT. Due to practical considerations, we approximate the Hiero search space with translation lattices. Fig. 4.1 illustrates the difference between NMT and our hybrid scheme with a simplified flow chart. In NMT, the decoder state at time j is used to build the posterior over the j-th target token via the embedding layer. The search strategy picks an index in the posterior vector and feeds back an embedding of it as input to the decoder network (Fig. 4.1a). In syntactically guided NMT, the posterior vector is modified. Entries which are not possible in the Hiero lattices are deleted, and lattice scores for other entries are added to the NMT scores. This may result in the search strategy selecting a different word which is then fed back to the NMT network.



(a) Decoder update in NMT.



(b) Decoder update in syntactically guided NMT.

Fig. 4.1 Update of the decoder hidden state in standard NMT and syntactically guided NMT. Context vectors and recursive connections are omitted in this figure for the sake of simplicity. Layers are annotated with their dimensionality in a standard RNNsearch setup.

More formally, NMT and Hiero scores are combined in a log-linear model. Recall the left-to-right factorization in NMT (Eq. 3.1):

$$P(\mathbf{y}|\mathbf{x}) = \prod_{j=1}^{J} P(y_j|y_1^{j-1},\mathbf{x}). \tag{4.1}$$

The hybrid system models the conditionals $P(y_j|y_1^{j-1},\mathbf{x})$ as follows (Stahlberg et al., 2016b, Eq. 5):

$$\log P(y_j|y_1^{j-1},\mathbf{x}) = \lambda_{Hiero} \log P_{Hiero}(y_j|y_1^{j-1},\mathbf{x}) + \lambda_{NMT} \begin{cases} \log P_{NMT}(y_j|y_1^{j-1},\mathbf{x}) & y_j \in \Sigma_{NMT} \\ \log P_{NMT}(\texttt{UNK}|y_1^{j-1},\mathbf{x}) & y_j \notin \Sigma_{NMT} \end{cases} \tag{4.2}$$

Note that the Hiero vocabulary is usually much larger than the NMT vocabulary Σ_{NMT}. If a Hiero translation contains a word not in the NMT vocabulary, the NMT model provides a score and updates its decoder state as for an unknown word (UNK).

In general, the combined score $P(y_j|y_1^{j-1},\mathbf{x})$ does not represent a probability distribution (i.e. does not sum up to 1) because the NMT UNK score may have been used more than once. We could renormalize $P(\cdot)$ by a factor which depends on the number of non-zero NMT OOVs in the current posterior vector. In practice, however, this does not yield any improvements. Another way to restore the probabilistic constraint is the *local softmax* (see Sec. 4.3.3).

4.3.1 Hiero Predictive Posteriors

As explained in Sec. 2.7.2, the Hiero decoder HiFST generates translation hypotheses for a source sentence $\mathbf{x}$ as weighted finite state transducers (WFSTs), or lattices, with weights in the tropical semiring. In a HiFST translation lattice T, each complete path $p \in \mathscr{P}(T)$ corresponds to a Hiero derivation D, and the path weight is the negative log of the Hiero score $P(D)$:

$$\forall p \in \mathscr{P}(T) : \exists D \in R^+ : \langle S,S\rangle \Rightarrow_D \langle \mathbf{x}, i[p]\rangle \wedge \omega[p] = -\log P(D) \tag{4.3}$$

R, S, $P(D)$, and $\Rightarrow_D$ are defined in Sec. 2.7.1, $\mathscr{P}(\cdot)$, $i[\cdot]$, and $\omega[\cdot]$ were introduced in Sec. 2.6.1. While this representation is correct with respect to the Hiero translation grammar and language model scores, having Hiero scores at the path level is not convenient for working with the NMT system. What we need are predictive probabilities in the form of Eq. 4.2.

The Hiero WFSAs are determinized and minimized with epsilon removal under the tropical semiring, and weights are pushed towards the initial state under the log semiring (cf. Sec. 2.6.1). The resulting transducer is stochastic in the log semiring, i.e. the log sum of the arc log probabilities leaving a state is 0 ($= \log 1$). In addition, because the WFSA is deterministic, there is a unique path leading to every state, which corresponds to a unique Hiero translation prefix.

Suppose a path p to a state $v \in V$ accepts the translation prefix y_1^{j-1}. An outgoing arc $e \in E_v$ from that state with symbol $y = i[e]$ has a weight that corresponds to the (negative log of the) conditional probability

$$-\log P_{Hiero}(y_j = y|y_1^{j-1},\mathbf{x}) = \omega(e). \tag{4.4}$$

This conditional probability is such that for a Hiero translation $\mathbf{y}$ with derivation D (i.e. $\langle S,S\rangle \Rightarrow_D \langle \mathbf{x},\mathbf{y}\rangle$)

$$P_{Hiero}(\mathbf{y}|\mathbf{x}) = \prod_{j=1}^{J} P_{Hiero}(y_j|y_1^{j-1},\mathbf{x}) \propto P(D). \tag{4.5}$$

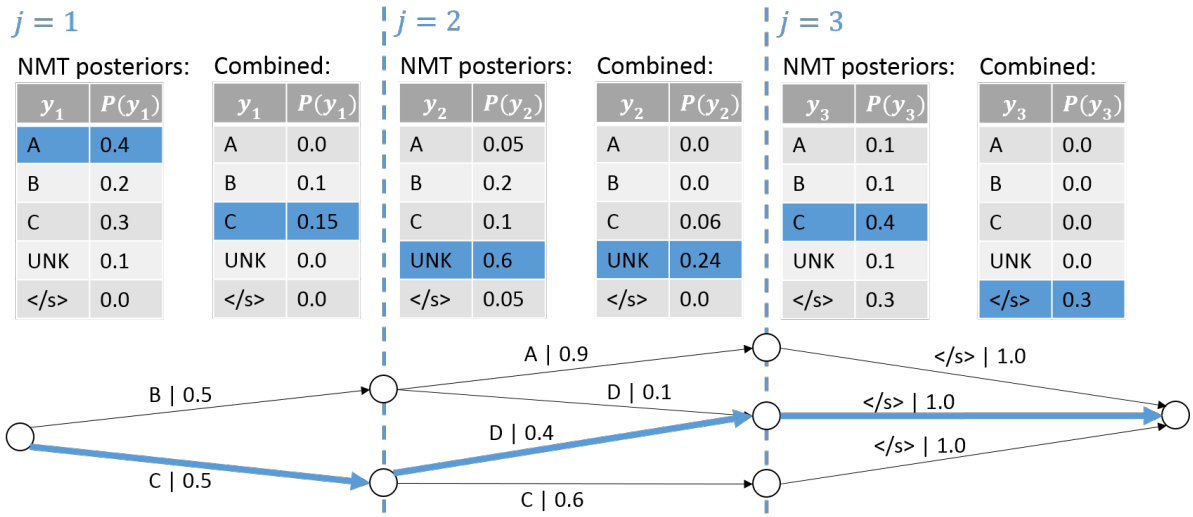


Fig. 4.2 Greedy decoding in syntactically guided NMT.

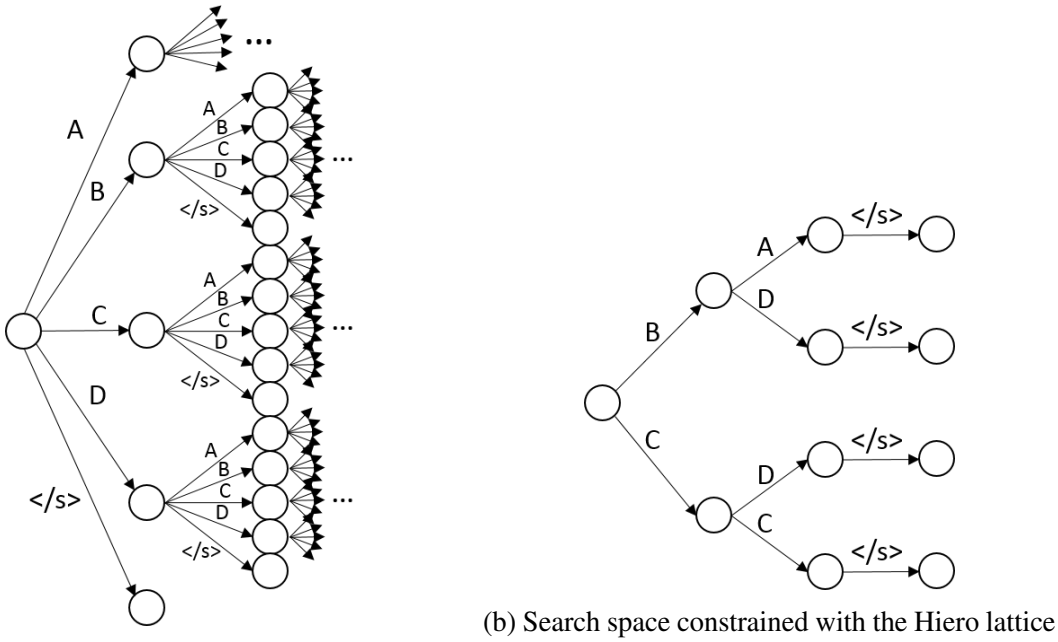
The Hiero WFSAs have been transformed so that their arc weights have the negative log of the conditional probabilities defined in Eq. 4.4. All the probability mass of this distribution is concentrated on the Hiero translation hypotheses. The complete translation and language model scores computed over the entire Hiero translations are pushed as far forward in the WFSAs as possible. This is commonly done for left-to-right decoding in speech recognition (Mohri and Riley, 2001).

4.3.2 FST-based Constrained NMT Decoding

Due to the form of Eq. 4.1 we can build up hypotheses from left to right on the target side. Thus, we can use constrained versions of the standard NMT greedy or beam search procedures from Sec. 3.7 for decoding. Fig. 4.2 illustrates how the greedy decoder traverses the search space. The initial state in the Hiero lattice has outgoing arcs labelled with symbols B and C. The arc scores are multiplied with the NMT posteriors at time $j = 1$, and the best symbol C is selected greedily. At $j = 2$, the Hiero lattice allows D and C. Since D is not in the NMT vocabulary, NMT provides a score for D through `UNK`. The final hypothesis is "C D </s>".

Syntactically guided NMT is related to conventional lattice rescoring, but there are a few important differences. First, in contrast to rescoring, our decoder does not visit the entire lattice exhaustively but runs approximate search (beam search) on the lattice. Exhaustive search with the neural model would be computationally far too expensive. Sec. 5.4.4 gives some insight in the number of search errors introduced by beam search.

The second important difference is that the hybrid search space is not in line with the conditional independence assumptions of the Hiero lattice anymore. The search space for



(a) Unconstrained NMT search space.

(b) Search space constrained with the Hiero lattice in Fig. 4.2.

Fig. 4.3 NMT unconstrained and constrained search spaces over the vocabulary $\{A, B, C, D, \text{</s>}\}$.

unconstrained NMT is a tree structure (Fig. 4.3a) due to the unlimited history length. Therefore, constraining it with an FST (Fig. 4.3b) again results in a tree although the original Hiero lattice from Fig. 4.2 is not a tree as continuations of the prefixes B D and C D are independent from those prefixes as the paths merge in the lattice. The realization that the constrained search space is a tree will later motivate us to apply standard tree traversal algorithms like depth-first search to neural sequence models (Sec. 5.4).

We will revisit the idea of constraining a neural decoder with an FST in Sec. 8.3 for our work on text normalization.

4.3.3 Experiments

Experimental Setup

We evaluate on the WMT15 test sets for English-German (En-De) and English-French (En-Fr). The En-De training set includes *Europarl v7*, *Common Crawl*, and *News Commentary v10*. Sentence pairs with sentences longer than 80 words or length ratios exceeding 2.4:1 were deleted, as were *Common Crawl* sentences from other languages (Shuyo, 2010). The En-Fr NMT system was trained on preprocessed data (Schwenk, 2014) used by previous work (Bahdanau et al., 2015; Jean et al., 2015a; Sutskever et al., 2014), but with truecasing like

	Train set		Dev set		test2014	
	En	De	En	De	En	De
# sentences	4.2M		6k		2.7k	
# word tokens	106M	102M	138k	138k	62k	59k
# unique words	647k	1.5M	13k	20k	9k	13k
OOV (Hiero)	0.0%	0.0%	0.8%	1.6%	1.0%	2.0%
OOV (NMT)	1.6%	5.5%	2.5%	7.5%	3.1%	8.8%

Table 4.2 Parallel texts and vocabulary coverage on En-De *news-test2014*.

	Train set		Dev set		test2014	
	En	Fr	En	Fr	En	Fr
# sentences	12.1M		6k		3k	
# word tokens	305M	348M	138k	155k	71k	81k
# unique words	1.6M	1.7M	14k	17k	10k	11k
OOV (Hiero)	0.0%	0.0%	0.6%	0.6%	0.4%	0.4%
OOV (NMT)	3.5%	3.8%	4.5%	5.3%	5.0%	5.3%

Table 4.3 Parallel texts and vocabulary coverage on En-Fr *news-test2014*.

our Hiero baseline. Following Jean et al. (2015a), we use *news-test2012* and *news-test2013* as a development set. The NMT vocabulary size is 50k for En-De and 30k for En-Fr, taken as the most frequent words in training (Jean et al., 2015a). Tab. 4.2 and 4.3 provide statistics and show the severity of the OOV problem for NMT.

The NMT systems are built using the Blocks framework (van Merriënboer et al., 2015) based on the Theano library (Bastien et al., 2012) with standard hyper-parameters (Bahdanau et al., 2015): the encoder and decoder networks consist of 1000 gated recurrent units (Cho et al., 2014b). The decoder uses a single maxout (Goodfellow et al., 2013b) output layer with the feed-forward attention model (Bahdanau et al., 2015).

The En-De Hiero system uses rules which encourage verb movement (de Gispert et al., 2010). The rules for En-Fr were extracted from the full data set available at the WMT'15 website using a shallow-1 grammar (de Gispert et al., 2010). 5-gram Kneser-Ney language models (KN-LM) for the Hiero systems were trained on WMT'15 parallel and monolingual data (Heafield et al., 2013).

Results

Tab. 4.4 compares syntactically guided NMT with other approaches in the literature. We use a beam size of 12. In En-De and in En-Fr, we find that our pure NMT systems perform similarly

Setup (related work)	BLEU En-De	BLEU En-Fr
Basic NMT	16.4	30.0
NMT-LV	17.0	33.4
+ UNK Replace	18.9	34.1
+ Reshuffle	19.4	34.6
+ Ensemble	21.6	37.2

(a) Results from Jean et al. (2015a, Tab. 2).

Setup (own work)	BLEU En-De	BLEU En-Fr
Basic NMT	16.3	30.4
Hiero	19.4	32.9
Hybrid (not tuned)	20.7	35.4
Hybrid (tuned)	21.4	36.3
+ Reshuffle	21.9	36.6

(b) Our own experiments.

Table 4.4 BLEU scores on *news-test2014* for En-De and En-Fr. NMT-LV refers to the RNNSEARCH-LV model (Jean et al., 2015a) for large output vocabularies.

(within 0.6 BLEU) to previously published word-based results with similar architectures (16.3 vs. 16.5 and 30.4 vs. 30.0).

In our untuned experiments, decoding is as described in Sec. 4.3.1, but with $\lambda_{Hiero} = 0$. The decoder searches through the Hiero lattice, ignoring the Hiero scores, but using Hiero word hypotheses in place of any UNKs that might have been produced by NMT. The results show that this is much more effective in fixing NMT OOVs than the 'UNK Replace' technique (Luong et al., 2015c) discussed in Sec. 3.8.1; this holds in both En-De and En-Fr. Using lattice MERT (Macherey et al., 2008) on the beam search n-best lists to optimize λ_{Hiero} and λ_{NMT} yields further gains in both En-Fr and En-De, suggesting that in addition to fixing UNKs, the Hiero predictive posteriors can be used to improve the NMT translation model scores.

Tab. 4.5 reports results of our En-De system with reshuffling[2] and tuning on *news-test2015*. BLEU scores are directly comparable to WMT'15 results.[3] By comparing row 3 to row 10, we see that constraining NMT to the search space defined by the Hiero lattices yields an improvement of +0.8 BLEU for single NMT. If we allow Hiero to fix NMT UNKs, we see a further +2.7 BLEU gain (row 11). The majority of gains come from fixing UNKs, but there is still improvement from the constrained search space for single NMT.

We next investigate the contribution of the Hiero system scores. We see that, once lattices are generated, the KN-LM contributes more to rescoring than the Hiero grammar scores (rows 12-14). Further gains can be achieved by adding a feed-forward neural language model with NPLM (Vaswani et al., 2013) (row 15). We observe that *n*-best list rescoring with NMT (Neubig et al., 2015) also outperforms both the Hiero and NMT baselines, although lattice rescoring gives the best results (row 9 vs. row 15). Lattice rescoring also uses far fewer node expansions per sentence. We report *n*-best rescoring speeds for rescoring each hypothesis separately, and a depth-first (DFS) scheme that efficiently traverses the *n*-best lists. Both these techniques are

[2]Reshuffling refers to randomly permuting the training set after each epoch.
[3]http://matrix.statmt.org/matrix/systems_list/1774

	Search space	Vocab.	NMT scores	Grammar scores	KN-LM scores	NPLM scores	# of node expansions per sen.	BLEU single	BLEU ensemble
1	Lattice	Hiero		✓	✓		–	21.1 (Hiero)	
2	Lattice	Hiero		✓	✓	✓	–	21.7 (Hiero)	
3	Unrestricted	NMT	✓				254.8	19.5	21.8
4	100-best	Hiero	✓				2,233.6 (DFS: 832.1)	22.8	23.3
5	100-best	Hiero	✓	✓	✓			22.9	23.4
6	100-best	Hiero	✓	✓	✓	✓		22.9	23.3
7	1000-best	Hiero	✓				21,686.2 (DFS: 6,221.8)	23.3	23.8
8	1000-best	Hiero	✓	✓	✓			23.4	23.9
9	1000-best	Hiero	✓	✓	✓	✓		23.5	24.0
10	Lattice	NMT	✓				243.3	20.3	21.4
11	Lattice	Hiero	✓				243.3	23.0	24.2
12	Lattice	Hiero	✓	✓			243.3	23.0	24.2
13	Lattice	Hiero	✓		✓		240.5	23.4	24.5
14	Lattice	Hiero	✓	✓	✓		243.9	23.4	24.4
15	Lattice	Hiero	✓	✓	✓	✓	244.3	24.0	24.4
16	Neural MT – UMontreal-MILA (Jean et al., 2015b)							22.8	25.2

Table 4.5 BLEU English-German *news-test2015* scores calculated with `mteval-v13a.pl`.

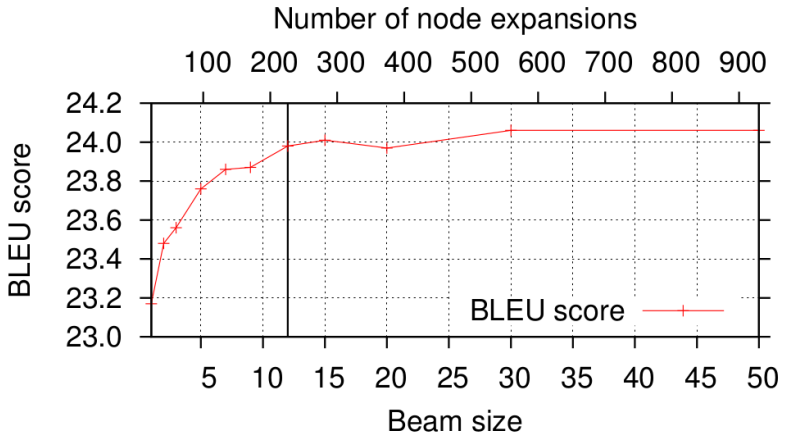


Fig. 4.4 Performance with NPLM over beam size on English-German *news-test2015*. A beam of 12 corresponds to row 15 in Tab. 4.5.

very slow compared to lattice rescoring. Fig. 4.4 shows that we can reduce the beam size from 12 to 5 with only a minor drop in BLEU. This is nearly 100 times faster than DFS over the 1000-best list.

Determinization	Minimization	Weight pushing	Sentences per second
✓			2.51
✓	✓		1.57
✓	✓	✓	1.47

Table 4.6 Time for lattice preprocessing operations on English-German *news-test2015*.

Cost of Lattice Preprocessing As described in Sec. 4.3.1, we applied determinization, minimization, and weight pushing to the Hiero lattices in order to work with probabilities. Tab. 4.6 shows that those operations are generally fast.[4]

Lattice Size For previous experiments we set the Hiero pruning parameters such that lattices had 8,510 nodes on average. Fig. 4.5 plots the BLEU score over the lattice size. We find that our combination approach works well on lattices of moderate or large size, but pruning lattices too heavily has a negative effect as they are then too similar to Hiero first best hypotheses. We note that lattice rescoring involves nearly as many node expansions as unconstrained NMT decoding. This confirms that the lattices at 8,510 nodes are already large enough for rescoring.

Local Softmax The vanilla architecture for word-based attentional NMT uses a softmax output layer over the full 50,000 words translation vocabulary. This layer increases the computational complexity significantly as it needs to sum over all 50,000 words for normalization. In our hybrid system we have the option of normalizing the NMT translation probabilities over the words on outgoing arcs from each state in the lattice rather than over the full 50,000 words. There are $\sim$4.5 arcs per state in our En-De'14 lattices, and so avoiding the full softmax could cause significant computational savings. We find this leads to only a modest 0.5 BLEU degradation: 21.45 BLEU in En-De'14, compared to 21.87 BLEU using NMT probabilities computed over the full vocabulary.

Modelling Errors vs. Search Errors In our En-De'14 experiments with $\lambda_{Hiero} = 0$ we find that constraining the NMT decoder to the Hiero lattices yields translation hypotheses with much lower NMT probabilities than unconstrained NMT decoding: hypotheses from unconstrained decoding are 8,300 times more likely according the NMT model (median) than hypotheses from the hybrid system. We conclude (tentatively) that pure NMT is not suffering only from search errors, but rather that the SMT-NMT hybrid discards some hypotheses ranked highly by the NMT model but lower in the evaluation metric, i.e. NMT assigns its highest scores to

[4]Testing environment: Ubuntu 14.04, Linux 3.13.0, single Intel® Xeon® X5650 CPU at 2.67 GHz

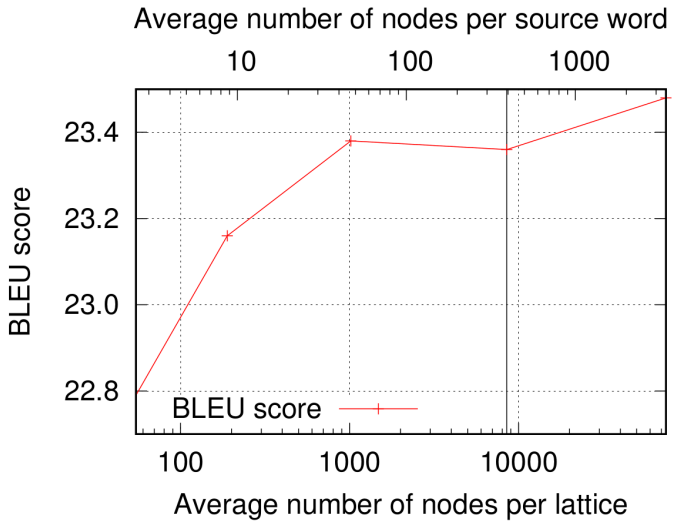


Fig. 4.5 The impact of Hiero lattice size on English-German *news-test2015*. 8,510 nodes per lattice corresponds to row 14 in Tab. 4.5.

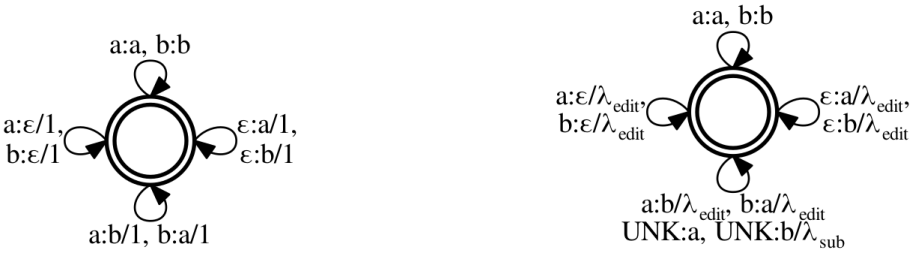
hypotheses that are not necessarily the best translations. Sec. 5.4.4 contains further discussions on NMT model and search errors.

4.4 UCAM@WMT16: Combination Via Edit Distance Transducer

This section describes our English-German submission to WMT16 (Stahlberg et al., 2016a). Syntactically guided NMT in the previous section can be seen as Hiero lattice rescoring with an NMT model, and thus enforces an exact match between the NMT and syntactic decoders. In general, this kind of hard restriction is best avoided when combining diverse systems (Frederking et al., 1994; Liu et al., 2009). For example, in speech recognition, ROVER (Fiscus, 1997) is a system combination approach based on a soft voting scheme.

We find that Hiero lattices generated by grammars extracted with the usual heuristics (Chiang, 2007) do not provide enough variety to explore the full potential of neural models. Therefore, we present a "soft" lattice-based combination scheme which uses standard operations on finite state transducers introduced in Sec. 2.6.[5] We will describe a procedure for combining NMT and Hiero that captures similarity under the edit distance (Levenshtein, 1966) and both the NMT and Hiero translation system scores. Thus, our method replaces the hard combination and gives the NMT decoder more freedom to diverge from the Hiero translations. We find that this loose coupling scheme is especially useful when using NMT ensembles.

[5] Sec. 4.5 presents another "soft" combination approach based on minimum Bayes-risk decoding.



(a) Standard edit distance transducer over the alphabet $\{a, b\}$.

(b) Modified edit distance transducer D over the alphabet $\{a, b, \text{UNK}\}$. 'a' is an NMT OOV.

Fig. 4.6 "Flower automata" for calculating edit distances.

4.4.1 The Edit Distance Transducer

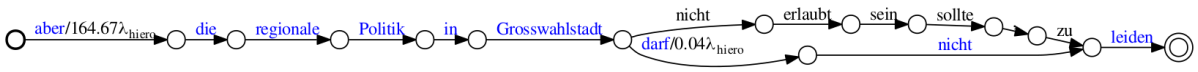
FST composition can be used together with a "flower automaton" to calculate the edit distance between two sequences (Mohri, 2003). The edit distance transducer shown in Fig. 4.6a transduces a sequence to another sequence over the alphabet $\{a, b\}$ and accumulates the number of edit operations via the transitions with cost 1. In our case, the input sequence corresponds to an NMT hypothesis which is to be combined with a Hiero hypothesis as output sequence. In contrast to lattice rescoring, where we require an exact match between NMT and Hiero (up to UNKs), our edit-distance-based scheme allows different hypotheses to be combined. We replaced the standard definition of the edit distance transducer (Mohri, 2003) by a finer-grained model designed to work well for combining NMT and Hiero. Instead of uniform costs, we lower the cost for UNK substitutions as we want to encourage substituting NMT UNKs by words in the Hiero translation. We distinguish between three types of edit operations.

- **Type I**: Substituting UNK with a word outside the NMT vocabulary is free.
- **Type II**: For substitutions of UNK with a word inside the NMT vocabulary we add the cost λ_{sub}.
- **Type III**: All other edit operations are penalized with cost λ_{edit} (and $\lambda_{edit} > \lambda_{sub}$).

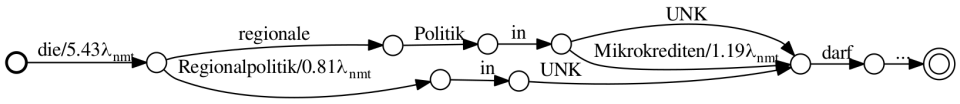
We will refer to the modified edit distance transducer as D. Fig. 4.6b shows D over the alphabet $\{a, b, \text{UNK}\}$, with 'a' being an NMT OOV.

4.4.2 Loose Coupling of Hiero and NMT

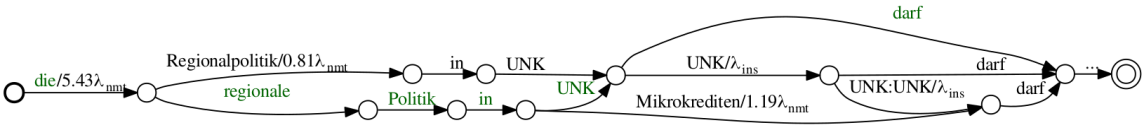
Our edit-distance-based scheme combines an NMT translation lattice N with a Hiero translation lattice H. Weights in N and H are scaled by λ_{NMT} and λ_{Hiero}, respectively. The similarity measure between NMT and Hiero translations is parametrized with λ_{ins}, λ_{edit}, and λ_{sub}. We



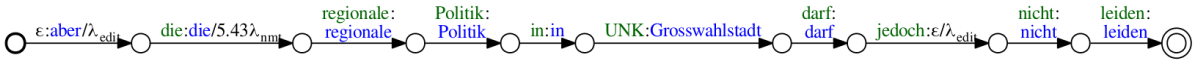
(a) Example Hiero lattice H.



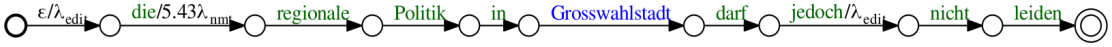
(b) Example NMT lattice N.



(c) Transducer with UNK insertion arcs: Replace$(N, \texttt{UNK}, U)$.



(d) Best path in the combined transducer C. Hiero scores are omitted in this figure.



(e) Projection of the best path: $\pi_{UNK}(\text{ShortestPath}(C))$. The final hypothesis is *die regionale Politik in Grosswahlstadt darf jedoch nicht leiden*.

Fig. 4.7 Combining Hiero and NMT via edit distance transducer.

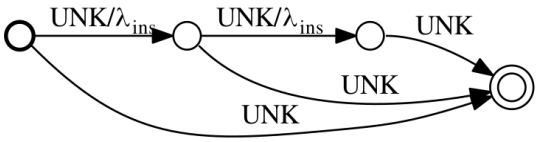


Fig. 4.8 UNK extension transducer U.

keep the various costs separated by using transducers with tropical sparse tuple vector semirings (Iglesias et al., 2015). Instead of single real-valued arc weights, this semiring uses vectors which can hold multiple features. The inner product of these vectors with a constant parameter vector determines the final weights on the arcs.[6] The sparse tuple vector semiring enables us to optimize the λ-parameters with the LMERT (Macherey et al., 2008) implementation of HiFST (de Gispert et al., 2010) on a development set.

Examples for H and N are shown in Fig. 4.7a and Fig. 4.7b. The shortest path in H containing the string *nicht erlaubt sein sollte zu* has grammatical and stylistic flaws but is complete, whereas there is a better path in N with an UNK. Our goal is to merge these two hypotheses by using the NMT translation in N with the UNK replaced by a word from the Hiero lattice H.

[6]For more information about this semiring see http://ucam-smt.github.io/tutorial/basictrans.html#lmert_veclats_tst

1. **Adding UNK insertions**. We found that often NMT produces an isolated UNK token, even if multiple tokens are required. Therefore, we allow extending a single UNK token to a sequence of up to three UNK tokens. This is realized by replacing UNK arcs in N with the transducer U shown in Fig. 4.8 using OpenFST's `Replace` operation. U connects the initial state with the final state directly with an UNK arc. Additionally, it contains UNK paths of lengths two and three with cost λ_{ins} and $2\lambda_{ins}$. Fig. 4.7c shows the result of the replace operation when applied to the example lattice N in Fig. 4.7b. We denote this operation as follows:

$$\text{Replace}(N, \texttt{UNK}, U) \tag{4.6}$$

2. **Composition with the edit distance transducer**. The next step finds the edit distances to the Hiero hypotheses as described in Sec. 4.4.1.

$$C := \text{Replace}(N, \texttt{UNK}, U) \circ D \circ H \tag{4.7}$$

3. **Shortest path**. The above operation generates very large lattices, and dumping all of them is not feasible. We could use disambiguation (Iglesias et al., 2015; Mohri and Riley, 2015) on the combined transducer C to find the best alignment for each unique NMT hypothesis. However, we only need the single shortest path in order to generate the combined translation.

$$\text{ShortestPath}(C) \tag{4.8}$$

4. **Projection**. A complete path in the transducer C has an NMT hypothesis on the input labels (marked green in Fig. 4.7d) and a Hiero hypothesis on the output labels (marked blue in Fig. 4.7d). Therefore, we can generate different translations from the best path in C. If we project the input labels on the output labels with OpenFST's `Project`, we obtain a hypothesis $\widehat{\mathbf{y}}_{NMT}$ in the NMT lattice N.[7]

$$\widehat{\mathbf{y}}_{NMT} = i[\text{ShortestPath}(C)] \tag{4.9}$$

However, $\widehat{\mathbf{y}}_{NMT}$ still contains UNKs. If we project on the input labels, we end up with the aligned Hiero hypothesis without UNKs (blue labels in Fig. 4.7d)

$$\widehat{\mathbf{y}}_{Hiero} = o[\text{ShortestPath}(C)] \tag{4.10}$$

[7]The projection functions $i \equiv \pi_1$ and $o \equiv \pi_2$ were defined in Sec. 2.6.

but we do not use the NMT translation directly. Therefore, we introduce a new projection function π_{UNK} which switches between preserving symbols on the input and output tapes:

$$\pi_{UNK}(e) = \begin{cases} i[e] & \text{if } i[e] \neq \texttt{UNK} \\ o[e] & \text{if } i[e] = \texttt{UNK} \end{cases}. \tag{4.11}$$

If the input label on an arc $e \in E$ is UNK, we write the output label over the input label. Otherwise, we write the input label over the output label. As shown in Fig. 4.7e, we obtain the NMT hypothesis, but the UNK is replaced by the matching word *Grosswahlstadt* from the Hiero lattice. Thus, the final combined translation is described by the following term:

$$\widehat{\mathbf{y}}_{comb} = \pi_{UNK}(\text{ShortestPath}(C)) \tag{4.12}$$

In general, the final hypothesis $\widehat{\mathbf{y}}_{comb}$ is a mix of an NMT and a Hiero hypothesis. We do not search for $\widehat{\mathbf{y}}_{comb}$ directly but for pairs of NMT and Hiero translations which optimize the individual model scores as well as the distance between them. Stated more formally, the shortest path in C yields a pair $(\widehat{\mathbf{y}}_{NMT}, \widehat{\mathbf{y}}_{Hiero})$ for which holds

$$\widehat{\mathbf{y}}_{NMT}, \widehat{\mathbf{y}}_{Hiero} = \underset{(\mathbf{y}_N, \mathbf{y}_H) \in N \times H}{\arg\min} \Big(d_{edit}(\mathbf{y}_N, \mathbf{y}_H) - \lambda_{NMT} \cdot \log P_{NMT}(\mathbf{y}_N|\mathbf{x}) - \lambda_{Hiero} \cdot \log P_{Hiero}(\mathbf{y}_H|\mathbf{x}) \Big) \tag{4.13}$$

where $d_{edit}(t_N, t_H)$ is the modified edit distance between $\mathbf{y}_N$ and $\mathbf{y}_H$ (according D and U), and $P_{NMT}(\mathbf{y}_H|\mathbf{x})$ and $P_{Hiero}(\mathbf{y}_H|\mathbf{x})$ are the NMT and Hiero translation scores as defined in Sec. 4.3.1. We arrive at a probabilistic interpretation of Eq. 4.13:

$$\widehat{\mathbf{y}}_{NMT}, \widehat{\mathbf{y}}_{Hiero} = \underset{(\mathbf{y}_N, \mathbf{y}_H) \in N \times H}{\arg\max} \Big(\mathrm{e}^{-d_{edit}(\mathbf{y}_N, \mathbf{y}_H)} \cdot P(\mathbf{y}_N, \mathbf{y}_H|\mathbf{x}) \Big). \tag{4.14}$$

with (assuming independence)

$$P(\mathbf{y}_N, \mathbf{y}_H|\mathbf{x}) := P_{NMT}(\mathbf{y}_N|\mathbf{x})^{\lambda_{NMT}} \cdot P_{Hiero}(\mathbf{y}_H|\mathbf{x})^{\lambda_{Hiero}}. \tag{4.15}$$

Eq. 4.14 suggests that we maximize the product of two quantities – the similarity between Hiero and NMT hypotheses and their joint probability. The FST operations allow to optimize over the set $N \times H$ efficiently. Note that the NMT lattice N is rather small in our case ($|N| \leq 20$) due to the small beam size used in NMT decoding. This makes it possible to solve Eq. 4.13 almost always without pruning.[8]

[8] We limit the Hiero lattices to a maximum of 100,000 nodes with OpenFST's `Prune` to remove the worst outliers.

Setup		test2014	test2015	test2016
Best in competition[9]		20.6	25.2	34.8
Hiero baseline		18.9	21.2	26.0
Single NMT	Pure NMT	17.5	19.6	23.2
	Syntactically guided NMT (Sec. 4.3)	21.2	23.5	28.7
	Edit distance based combination	21.7	24.1	28.6
Ensemble NMT	Pure NMT	19.4	21.7	25.4
	Syntactically guided NMT (Sec. 4.3)	21.9	24.6	29.7
	Edit distance based combination	**22.9**	**25.7**	**31.3**

Table 4.7 English-German lower-cased BLEU scores calculated with `mteval-v13a.pl`.

4.4.3 Experiments

Tab. 4.7 reports performance on *news-test2014*, *news-test2015*, and *news-test2016*. Syntactically guided NMT (Sec. 4.3) outperforms both NMT and Hiero baselines significantly on all test sets. We see further improvements of between +0.7 BLEU (*news-test2014*) and +1.1 BLEU (*news-test2015*) by using NMT ensembles rather than single NMT. However, these gains are rather small considering the improvements from using ensembles for the (pure) NMT baseline (between +1.9 BLEU and +2.2 BLEU). The edit-distance-based combination scheme makes better use of the ensembles. We report 31.3 BLEU on *news-test2016*, which in the English-German WMT'16 evaluation is among the best systems (within 0.1 BLEU) which do not use back-translation (Sennrich et al., 2016b).

The loose combination procedure we propose based on the edit distance is non-trivial. It is not immediately clear how the gains arise as the final scores are mixtures between edit distance costs, NMT scores, and Hiero scores. In the remainder we will try to provide some insight. Unless stated otherwise, we report investigations into the Hiero + NMT 8-system ensemble which yields the best results in Tab. 4.7.

First, we focus on the projection function $\pi_{UNK}(\cdot)$ which switches between preserving the input and output label at the UNK symbol to produce the combined translation $\widehat{\mathbf{y}}_{comb}$ (Eq. 4.12). As explained in Sec. 4.4.2, we can use OpenFST's `Project` operation to fetch the NMT and Hiero hypotheses $\widehat{\mathbf{y}}_{NMT}$ and $\widehat{\mathbf{y}}_{Hiero}$ which have been used to produce the combined translation (Eq. 4.9 and 4.10). Tab. 4.8 shows that the hypotheses that are aligned in the final transducer are often not the 1-best translations of any of the baseline systems. Remarkably, using the $\widehat{\mathbf{y}}_{Hiero}$ translations results in 30.4 BLEU, which is a very substantial improvement over the baseline Hiero system (26.0 BLEU). Note that this BLEU score is achieved with hypotheses from the original Hiero lattice H but weighted in combination with the NMT scores and the edit distance.

[9] http://matrix.statmt.org/

Method	BLEU
NMT baseline: ShortestPath(N)	25.4
Hiero baseline: ShortestPath(H)	26.4
NMT hypothesis used for combination: $\widehat{\mathbf{y}}_{NMT}$	26.7
Hiero hypothesis used for combination: $\widehat{\mathbf{y}}_{Hiero}$	30.4
Combined translation: $\widehat{\mathbf{y}}_{comb}$	31.3

Table 4.8 Projection methods on *news-test2016* with NMT 8-ensemble.

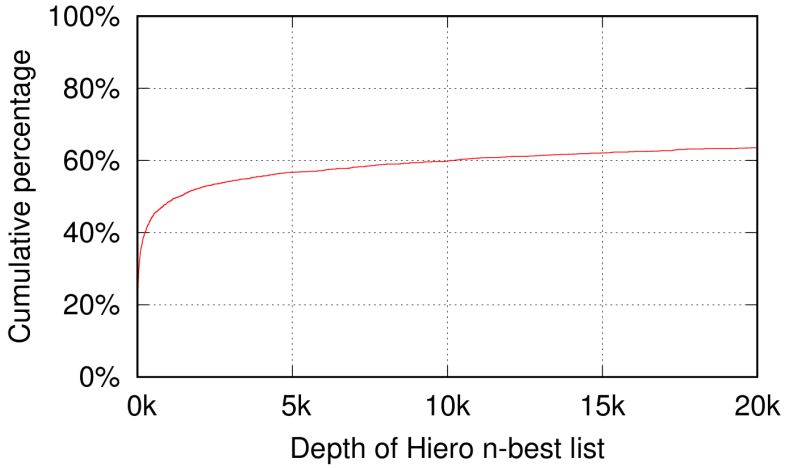


Fig. 4.9 Percentage of $\widehat{\mathbf{y}}_{Hiero}$ hypotheses found in the baseline Hiero n-best list.

However, these selected paths are often given very low scores by Hiero: in only 8.6% of the sentences is the Hiero hypothesis left unchanged. If we look for $\widehat{\mathbf{y}}_{Hiero}$ in the Hiero n-best list, we find that even very deep 20,000-best lists contain only 63.5% of the Hiero hypotheses which were selected by the combination scheme (Fig. 4.9). This indicates the benefit in using lattice-based approaches over n-best lists.

Next, we investigate the distance measure between NMT and Hiero translations, which is realized with the UNK insertion transducer U and the modified edit distance transducer D (Sec. 4.4.2). Tab. 4.9 shows that UNK insertions are relatively rare compared to the edit operations of types II and III allowed by D. The average edit distance between NMT and Hiero disregarding UNKs on the best path (type III) is 1.74. In 61.7% of the cases the input and output labels differ not only at UNK – i.e. in only 38.3% of the sentences do we have an exact match between NMT and Hiero. We note that UNK is often replaced with an NMT in-vocabulary word (55.9% of the sentences). It seems that NMT often produces an UNK even if a better word is in the NMT vocabulary. This could be due to the over-representation of UNK in the NMT training corpus.

Distance measure component	Avg. number per sentence	Percentage of affected sentences
UNK insertions (U)	0.16	12.9%
UNK→non-OOV substitutions (Type II)	1.34	55.9%
Other edit operations (Type III)	1.74	61.7%

Table 4.9 Breakdown of the distances measured between NMT and Hiero along the shortest path in C on *news-test2016*.

Vocabulary size	Pure NMT BLEU	Pure NMT # of UNKs	NMT+Hiero BLEU
10,000	18.9	18.0%	28.1
30,000	21.6	16.3%	28.8
50,000	23.2	9.1%	28.6
60,000	22.9	9.9%	28.5

Table 4.10 BLEU scores on *news-test2016* for different vocabulary sizes (single NMT). Each individual NMT system is combined with Hiero as described in Sec. 4.4.2.

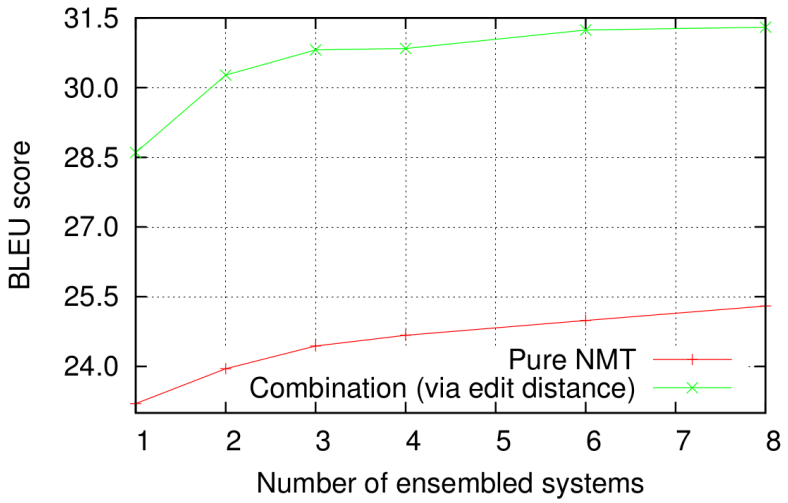


Fig. 4.10 BLEU score over the number of systems in the ensemble on *news-test2016*.

To study the effectiveness of our edit distance transducer based combination scheme in correcting NMT UNKs, we trained individual NMT systems with vocabulary sizes between 10,000 and 60,000. Tab. 4.10 shows that nearly one in six tokens (16.3%) produced by our pure NMT system with a vocabulary size of 30,000 are UNKs. Increasing the NMT vocabulary to 50k or 60k does improve pure NMT very significantly, but results show that these improvements are already captured by the combination scheme with Hiero. As in the

literature, we see large variation in performance over individual NMT systems even with the same vocabulary size (Sennrich et al., 2016c), which could explain the small performance drop when increasing the vocabulary size from 50k to 60k.

One important practical issue for system building is the number of systems to be ensembled as training each individual NMT system takes a significant amount of time. Fig. 4.10 indicates that even for 8-ensembles the gains for pure NMT do not seem to saturate. The combination with Hiero via edit distance transducer also greatly benefits from using ensembles, but most of the gains are gotten with fewer systems.

4.5 MBR-based Hybrid Machine Translation

In this section, we present our scheme to combine NMT with SMT by borrowing ideas from linearized lattice minimum Bayes-risk ((L)MBR) decoding (Goel et al., 2000; Kumar and Byrne, 2004; Tromble et al., 2008) for SMT. The NMT score is combined with the Bayes-risk of the translation according the SMT lattice. Similarly to our edit-distance-based method in Sec. 4.4, this makes MBR-based combination much more flexible than n-best list or lattice rescoring as the neural decoder is not restricted to the SMT search space.

Lattice minimum Bayes-risk (LMBR) decoding has been applied successfully to translation lattices in traditional SMT to improve translation performance of a single system (Blackwood et al., 2010; Kumar and Byrne, 2004; Tromble et al., 2008). Minimum Bayes-risk decoding is also a very powerful framework for combining diverse systems (de Gispert et al., 2009; Sim et al., 2007). However, we argue that MBR-based methods in their present form are not well-suited for NMT because of the following reasons:

- Previous approaches work well with rich lattices and diverse hypotheses. However, NMT decoding usually relies on beam search with a limited beam and thus produces very narrow lattices (Sec. 3.7.7).
- NMT decoding is computationally expensive. Therefore, it is difficult to collect the statistics needed for risk calculation for NMT.
- The Bayes-risk in SMT is usually defined for complete translations. Therefore, the risk computation needs to be restructured in order to integrate it in an NMT decoder which builds up hypotheses from left to right.

To address these challenges, we use a special loss function which is computationally tractable as it avoids using NMT scores for risk calculation. We show how to reformulate the original LMBR decision rule for using it in a word-based NMT decoder which is not restricted

to an n-best list or a lattice. We report significant gains over lattice rescoring on several data sets for both single and ensembled NMT, and show that the MBR decoder produces entirely new hypotheses far beyond simply rescoring the SMT search space or fixing UNKs in the NMT output. We also demonstrate the effectiveness of our method for subword-unit-based NMT. Our MBR-based approach has been proven useful even for practical industry-level MT (Iglesias et al., 2018).

4.5.1 Combining NMT and SMT by Minimizing the Lattice Bayes-risk

We propose to collect statistics for MBR from a potentially large translation lattice generated with SMT, and use the n-gram posteriors as additional score in NMT decoding. The LMBR decision rule used by Tromble et al. (2008) has the form

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}\in\mathscr{Y}_h} \underbrace{\left(\Theta_0|\mathbf{y}| + \sum_{\mathbf{u}\in\mathscr{N}} \Theta_{|\mathbf{u}|}\#_{\mathbf{u}}(\mathbf{y})P(\mathbf{u}|\mathscr{Y}_e)\right)}_{:=E_{SMT}(\mathbf{y})} \tag{4.16}$$

where $\mathscr{Y}_h$ is the *hypothesis space* of possible translations, $\mathscr{Y}_e$ is the *evidence space* for computing the Bayes-risk, and $\mathscr{N}$ is the set of all n-grams in $\mathscr{Y}_e$ (typically, $n = 1\ldots4$). In this section, our evidence space $\mathscr{Y}_e$ is a translation lattice generated with SMT. The function $\#_{\mathbf{u}}(\mathbf{y})$ counts how often n-gram $\mathbf{u}$ occurs in translation $\mathbf{y}$. $P(\mathbf{u}|\mathscr{Y}_e)$ denotes the path posterior probability of $\mathbf{u}$ in $\mathscr{Y}_e$ (See Sec. 4.5.3). Our aim is to integrate these n-gram posteriors into the NMT decoder since they correlate well with the presence of n-grams in reference translations (de Gispert et al., 2013). We call the quantity to be maximized the *evidence* $E_{SMT}(\mathbf{y})$ which corresponds to the (negative) Bayes-risk which is normally minimized in MBR decoding. We emphasize that this risk can be computed for any translation hypothesis and not only those produced by the SMT system.

Recall Eq. 3.1 that states that NMT assigns a probability to a translation $\mathbf{y}$ of source sentence $\mathbf{x}$ via a left-to-right factorization based on the chain rule:

$$P_{NMT}(\mathbf{y}|\mathbf{x}) = \prod_{j=1}^{|\mathbf{y}|} P_{NMT}(y_j|y_1^{j-1},\mathbf{x}). \tag{4.17}$$

$P_{NMT}(\cdot)$ can also represent an ensemble of NMT systems in which case the scores of the individual systems are multiplied together to form a single distribution. Applying the LMBR decision rule in Eq. 4.16 directly to NMT would involve computing $P_{NMT}(\mathbf{y}|\mathbf{x})$ for all translations in the evidence space. In case of LMBR this is equivalent to rescoring the entire translation lattice exhaustively with NMT. However, this is computationally intractable even for small lattices.

Therefore, we propose to calculate the Bayes-risk over SMT translation lattices using only pure SMT scores, and bias the NMT decoder towards low-risk hypotheses. Our final combined decision rule is

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} \Big(E_{SMT}(\mathbf{y}) + \lambda \log P_{NMT}(\mathbf{y}|\mathbf{x}) \Big). \tag{4.18}$$

If $\mathbf{y}$ contains a word not in the NMT vocabulary, the NMT model provides a score and updates its decoder state as for an unknown word. As we note, $E_{SMT}(\mathbf{y})$ can be computed even if $\mathbf{y}$ is not in the SMT lattice. Therefore, Eq. 4.18 can be used to generate translations outside the SMT search space.

4.5.2 Left-to-right Decoding

We explained in Sec. 3.7 how NMT builds up hypotheses from left to right using the factorization in Eq. 4.17. In contrast, our definition of the *evidence* in Eq. 4.16 contains a sum over the (unordered) set of all n-grams. However, we can rewrite our objective function in Eq. 4.18 in a way which makes it easy to use with beam search.

$$\begin{aligned} E_{SMT}(\mathbf{y}) + \lambda \log P_{NMT}(\mathbf{y}|\mathbf{x}) = \;& \Theta_0|\mathbf{y}| + \sum_{\mathbf{u}\in\mathcal{N}} \Theta_{|\mathbf{u}|} \#_{\mathbf{u}}(\mathbf{y}) P(\mathbf{u}|\mathscr{Y}_e) + \lambda \sum_{j=1}^{|\mathbf{y}|} \log P_{NMT}(y_j|y_1^{j-1},\mathbf{x}) \\ = \;& \sum_{j=1}^{|\mathbf{y}|} \Big(\Theta_0 + \sum_{n=1}^{4} \Theta_n P(y_{j-n}^{j}|\mathscr{Y}_e) + \lambda \log P_{NMT}(y_j|y_1^{j-1},\mathbf{x}) \Big) \end{aligned} \tag{4.19}$$

for n-grams up to order 4. This form lends itself naturally to beam search: at each time step, we add to the previous partial hypothesis score both the log-likelihood of the last token according the NMT model, and the partial MBR gains from the current n-gram history. Note that this is similar to applying an interpolated language model based on n-gram posteriors extracted from the SMT lattice. In the remainder, we will refer to decoding according Eq. 4.19 as *MBR-based* NMT.

4.5.3 Efficient n-gram Posterior Calculation

The risk computation in our approach is based on posterior probabilities $P(\mathbf{u}|\mathscr{Y}_e)$ for n-grams $\mathbf{u}$ which we extract from the SMT translation lattice $\mathscr{Y}_e$. $P(\mathbf{u}|\mathscr{Y}_e)$ is defined as the sum of the path probabilities $P_{SMT}(\cdot)$ of paths in $\mathscr{Y}_e$ containing $\mathbf{u}$ (Blackwood et al., 2010, Eq. 2):

$$P(\mathbf{u}|\mathscr{Y}_e) = \sum_{\mathbf{y}\in\{\mathbf{y}\in\mathscr{Y}_e:\#_{\mathbf{u}}(\mathbf{y})>0\}} P_{SMT}(\mathbf{y}|\mathbf{x}). \tag{4.20}$$

We use the framework of Blackwood et al. (2010) based on n-gram mapping and path counting transducers to efficiently pre-compute all non-zero values of $P(\mathbf{u}|\mathscr{Y}_e)$. Complete enumeration of all n-grams in a lattice is usually feasible even for very large lattices (Blackwood et al., 2010). Additionally, for all these n-grams $\mathbf{u}$, we smooth $P(\mathbf{u}|\mathscr{Y}_e)$ by mixing it with the uniform distribution to flatten the distribution and increase the offset to n-grams which are not in the lattice.

An interesting alternative way to obtain these n-gram probabilities was proposed by Zhang et al. (2018c) who used a search engine rather than an SMT system to retrieve likely n-grams.

4.5.4 Relation to MBR Decoding

For completeness we note that Eq. 4.18 can be interpreted as instance of the general form of MBR decoding for machine translation (Kumar and Byrne, 2004)

$$\hat{\mathbf{y}} = \arg\min_{\mathbf{y}\in\mathscr{Y}_h} \sum_{\mathbf{y}'\in\mathscr{Y}_e} L'(\mathbf{y}',\mathbf{y})P(\mathbf{y}'|\mathbf{x}) \tag{4.21}$$

by defining a modified loss function $L'(\cdot,\cdot)$ as follows:

$$L'(\mathbf{y}',\mathbf{y}) = \begin{cases} L(\mathbf{y}',\mathbf{y}) & \mathbf{y}' \neq \mathbf{y} \\ L(\mathbf{y}',\mathbf{y}) - \lambda \frac{\log P_{NMT}(\mathbf{y}|\mathbf{x})}{P(\mathbf{y}|\mathbf{x})} & \mathbf{y}' = \mathbf{y} \end{cases} \tag{4.22}$$

where $L(\cdot,\cdot)$ denotes the loss function commonly used in MBR for machine translation (Tromble et al., 2008).[10] Eq. 4.18 is then an approximation for Eq. 4.21:

$$\begin{aligned} \arg\min_{\mathbf{y}\in\mathscr{Y}_h} \sum_{\mathbf{y}'\in\mathscr{Y}_e} L'(\mathbf{y}',\mathbf{y})P(\mathbf{y}'|\mathbf{x}) &\overset{\text{Eq. 4.22}}{=} \arg\min_{\mathbf{y}\in\mathscr{Y}_h} \Big(\sum_{\mathbf{y}'\in\mathscr{Y}_e} L(\mathbf{y}',\mathbf{y})P(\mathbf{y}'|\mathbf{x}) - \lambda \log P_{NMT}(\mathbf{y}|\mathbf{x}) \Big) \\ &\overset{\text{Eq. 4.16}}{\approx} \arg\max_{\mathbf{y}} \Big(E(\mathbf{y}) + \lambda \log P_{NMT}(\mathbf{y}|\mathbf{x}) \Big). \end{aligned} \tag{4.23}$$

4.5.5 Subword-level MBR

Character-based or subword-unit-based NMT (Sec. 3.8.3) tries to overcome the limited vocabulary in word-based NMT. Hybrid systems offer an alternative way to fix NMT OOVs. However, our MBR-based method can also be used with subword-unit-based NMT. First, we construct a finite state transducer (FST) which maps word sequences to BPE sequences. Then, we convert

[10] Formally, division by zero in Eq. 4.22 can be avoided by smoothing $P(\cdot)$. In practice, however, we avoid this problem by using Eq. 4.18 directly.

the word-based SMT lattices to BPE-based lattices by composing them with the mapping transducer and projecting the output tape using standard OpenFST operations (Allauzen et al., 2007). The converted lattices are used for extracting n-gram posteriors as described in the previous sections. Note that even though the n-grams are on the BPE level, their posteriors are computed from word-level SMT translation scores.

4.5.6 Experiments

Experimental Setup

We test our approach on English-German (En-De) and Japanese-English (Ja-En). For En-De, we use the WMT *news-test2014* (the filtered version) as a development set, and keep *news-test2015* and *news-test2016* as test sets. For Ja-En, we use the ASPEC corpus (Nakazawa et al., 2016) to be strictly comparable to the evaluation done in the Workshop of Asian Translation (WAT).

The NMT systems are as described in Sec. 4.3.3 with vocabulary sizes of 30K for Ja-En and 50K for En-De. We use the coverage penalty proposed by Wu et al. (2016b) (Sec. 3.10.1) to improve the length and coverage of translations. Our final ensembles combine five (En-De) to six (Ja-En) independently trained NMT systems.

As in the previous sections we use the hierarchical FST-based SMT system HiFST (Sec. 2.7.2) for En-De. In Ja-En we use Travatar (Neubig, 2013), an open-source tree-to-string system. We provide the system with Japanese trees obtained using the Ckylark parser (Oda et al., 2015) and train it on high-quality alignments as recommended by Neubig and Duh (2014). This system, which reproduces the results of the best submission in WAT 2014 (Neubig, 2014), is used to create a 10K-best list of hypotheses, which we convert into determinized and minimized FSAs for our work. Our Ja-En NMT models are trained on the same 500K training samples as the Travatar baseline.

The parameter λ is tuned by optimizing the BLEU score on the validation set, and we set $\Theta_i = 1$ ($i = 0, \ldots, 4$). Using the BOBYQA algorithm (Powell, 2009) or lattice MERT (Macherey et al., 2008) to optimize the Θ-parameters independently did not yield improvements. We set the beam size to 20 for En-De and 12 for Ja-En.

Results

Our results are summarized in Tab. 4.11 and 4.12. Our approach outperforms both single NMT and SMT baselines by up to 3.4 BLEU for En-De and 2.8 BLEU for Ja-En. Ensembling yields further gains across all test sets both for the NMT baselines and our MBR-based hybrid systems. We see substantial gains from our MBR-based method over lattice rescoring for both single

Setup		news-test2014	news-test2015	news-test2016
SMT baseline (de Gispert et al., 2010, HiFST)		18.9	21.2	26.0
Single NMT (word)	Pure NMT	17.7	19.6	23.1
	100-best rescoring	20.6	22.5	27.5
	Lattice rescoring	21.6	23.8	**29.6**
	LMBR	**22.0**	**24.6**	29.5
5-Ensemble NMT (word)	Pure NMT	19.4	21.8	25.4
	100-best rescoring	21.0	23.3	28.6
	Lattice rescoring	22.1	24.2	30.2
	LMBR	**22.8**	**25.4**	**30.8**
Single NMT (BPE)	Pure NMT	19.6	21.9	24.6
	Lattice rescoring	21.5	24.0	**29.6**
	LMBR	**21.7**	**24.1**	28.6
3-Ensemble NMT (BPE)	Pure NMT	21.0	23.4	27.0
	Lattice rescoring	21.7	24.2	**30.0**
	LMBR	**22.3**	**24.9**	29.2

Table 4.11 English-German lower-cased BLEU scores calculated with `mteval-v13a.pl`. **LMBR** as described in Sec. 4.5.1.

Setup		Dev	Test
SMT baseline (Neubig, 2013, Travatar)		19.5	22.2
Single NMT (word)	Pure NMT	20.3	22.5
	10k-best rescoring	22.2	24.5
	LMBR	**22.4**	**25.2**
6-Ensemble NMT (word)	Pure NMT	22.6	25.0
	10k-best rescoring	22.4	25.4
	LMBR	**23.9**	**26.5**
Single NMT (BPE)	Pure NMT	20.8	23.5
	10k-best rescoring	21.9	24.6
	LMBR	**23.0**	**25.4**
3-Ensemble NMT (BPE)	Pure NMT	23.3	25.9
	10k-best rescoring	22.6	25.1
	LMBR	**24.1**	**26.7**

Table 4.12 Japanese-English cased BLEU scores calculated with Moses' `multi-bleu.pl`. **LMBR** as described in Sec. 4.5.1.

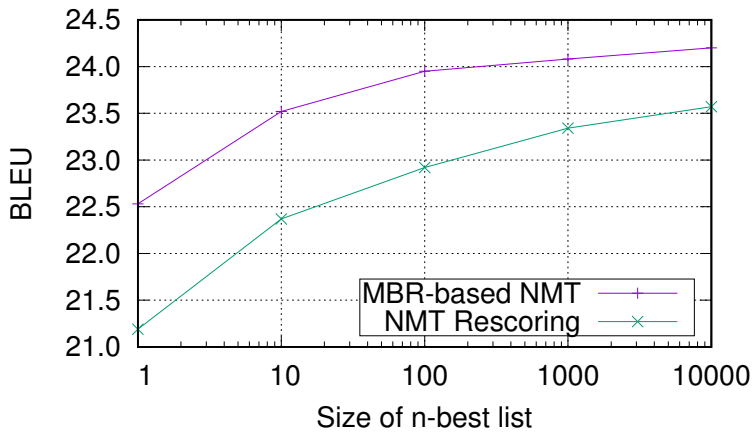


Fig. 4.11 Performance over *n*-best list size on English-German *news-test2015*.

and ensembled NMT on all test sets and language pairs except En-De *news-test2016*. On Ja-En, we report 26.7 BLEU, second to only one system (as of February 2017) that uses a number of techniques such as minimum risk training and a much larger vocabulary size which could also be used in our framework.

Our word-level NMT baselines suffer from their limited vocabulary since we do not apply post-processing techniques like UNK-replace (Luong et al., 2015c). Therefore, NMT with subword units (BPE) consistently outperforms them by a large margin. Lattice rescoring and MBR yield large gains for both BPE-based and word-based NMT. However, the performance difference between BPE- and word-level NMT diminishes with lattice rescoring and MBR decoding: rescoring with NMT often performs on the same level for both words and subword units, and MBR-based NMT is often even better with a word-level NMT baseline. This indicates that subword units are often not necessary when the hybrid system has access to a large word-level vocabulary like the SMT vocabulary.

Note that the BPE lattice rescoring system is constrained to produce words in the output vocabulary of the syntactic SMT system and is prevented from inventing new target language words out of combinations of subword units. MBR imposes a soft version of such a constraint by biasing the BPE-based system towards words in the SMT search space.

The hypotheses produced by our MBR-based method often differ from the translations in the baseline systems. For example, 77.8% of the translations from our best MBR-based system on Ja-En cannot be found in the SMT 10K-best list, and 78.0% do not match the translation from the pure NMT 6-ensemble.[11] This suggests that our MBR decoder is able to produce entirely new hypotheses, and that our method has a profound effect on the translations which goes beyond rescoring the SMT search space or fixing UNKs in the NMT output.

[11] Up to NMT OOVs.

Tab. 4.11 also shows that rescoring is sensitive to the size of the n-best list or lattice: rescoring the entire lattice instead of a 100-best list often yields a gain of 1 full BLEU point. In order to test our MBR-based method on small lattices, we compiled n-best lists of varying sizes to lattices and extracted n-gram posteriors from the reduced lattices. Fig. 4.11 shows that the n-best list size has an impact on both methods. Rescoring a 10-best list already yields a large improvement of 1.2 BLEU. However, the hypotheses are still close to the SMT baseline. The MBR-based approach can make better use of small n-best lists as it does not suffer this restriction. MBR-based combination on a 10-best list performs on about the same level as rescoring a 10,000-best list which demonstrates a practical advantage of MBR over rescoring.

4.6 UCAM@WMT18: MBR-based System Combination of Multiple Models

The previous section introduced our MBR-based scheme to combine NMT and SMT without constraining NMT to the SMT search space. For our submission to the WMT18 evaluation campaign (Bojar et al., 2018) we generalized our scheme to more than two systems. We discussed various NMT architectures in Ch. 3, and compared prominent examples of each class in Sec. 3.6.6. In the spirit of Chen et al. (2018b), we devoted our WMT18 submission to an empirical exploration of the three most commonly used architectures: recurrent, convolutional, and self-attention-based models like the Transformer. Our experiments shown next suggest that self-attention is the superior architecture on the tested language pairs, but it can still benefit from MBR model combination with the other two. As in the previous section we also report gains from MBR-based combination with a phrase-based SMT system. We found this particularly striking as the SMT baselines are often more than 10 BLEU points below our strongest neural models. Our final submission ranked second in terms of BLEU score in the WMT18 evaluation campaign on English-German and German-English, and outperforms all other systems on a variety of linguistic phenomena on German-English (Macketanz et al., 2018).

4.6.1 Multiple System Combination

In the previous section, we combined SMT and NMT in a hybrid system with a minimum Bayes-risk (MBR) formulation which has been proven useful even for practical industry-level MT (Iglesias et al., 2018). Our combination scheme for multiple systems is a generalization of this approach to more than two systems. Suppose we want to combine q models $\mathscr{M}_1, \ldots, \mathscr{M}_q$. We first divide the models into two groups by selecting a p with $1 \leq p \leq q$. We refer to scores from the first group $\mathscr{M}_1, \ldots, \mathscr{M}_p$ as *full posterior* scores and from the second group

$\mathscr{M}_{p+1}, \ldots, \mathscr{M}_q$ as *MBR-based* scores. Full posterior models contribute to the combined score with their complete posterior of the full translation. In contrast, models in the second group only provide the evidence space for estimating the probability of n-grams occurring in the translation. Full-posterior models need to assign scores via the standard left-to-right factorization (Eq. 3.1) of neural sequence models:

$$\log P(\mathbf{y}|\mathbf{x}, \mathscr{M}_k) = \sum_{j=1}^{|\mathbf{y}|} \log P(y_j|y_1^{j-1}, \mathbf{x}, \mathscr{M}_k) \tag{4.24}$$

for all $k \leq p$. For example, all left-to-right neural models in this work can be used as full posterior models, but the right-to-left models (Sec. 3.7.5) and SMT cannot. We combine full-posterior scores log-linearly, and bias the combined score $S(\mathbf{y}|\mathbf{x})$ towards low-risk hypotheses with respect to the MBR-based group as suggested in Sec. 4.5.2, Eq. 4.19:[12]

$$S(\mathbf{y}|\mathbf{x}) = \sum_{j=1}^{|\mathbf{y}|} \Bigg(\underbrace{\sum_{k=1}^{p} \lambda_k \log P(y_j|y_1^{j-1}, \mathbf{x}, \mathscr{M}_k)}_{\text{Full posterior}} + \underbrace{\sum_{l=p+1}^{q} \lambda_l \sum_{n=1}^{4} P(y_{j-n}^{j}|\mathbf{x}, \mathscr{M}_l)}_{\text{MBR-based } n\text{-gram scores}} \Bigg) \tag{4.25}$$

where $\lambda_1, \ldots, \lambda_q$ are interpolation weights. Eq. 4.25 also describes how to use beam search in this framework as hypotheses can be built up from left to right due to the outer sum over time steps. The MBR-based models contribute via the probability $P(y_{j-n}^{j}|\mathbf{x}, \mathscr{M}_l)$ of an n-gram y_{j-n}^{j} given the source sentence $\mathbf{x}$. Posteriors in this form are commonly used for MBR decoding in SMT (Kumar and Byrne, 2004; Tromble et al., 2008), and can be extracted efficiently as outlined in Sec. 4.5.3. For our neural models we run beam search with beam size 15 and compute posteriors over the 15-best list.

Note that our generalization to more than two systems can still be seen as instance of the original scheme from Sec. 4.5 by viewing the first group $\mathscr{M}_1, \ldots, \mathscr{M}_p$ as ensemble and the evidence space from the second group $\mathscr{M}_{p+1}, \ldots, \mathscr{M}_q$ as mixture model.

The performance of our system combinations depends on the correct calibration of the interpolation weights $\lambda_1, \ldots, \lambda_q$. We first tried to use n-best or lattice MERT (Macherey et al., 2008; Och, 2003) to find interpolation weights, but these techniques were not effective in our setting, possibly due to the lack of diversity and depth in n-best lists from standard beam search. Therefore, we tune on the first best translation using Powell's method (Powell, 1964) with a

[12] Eq. 4.25 differs from Eq. 4.19 in that we do not use a word penalty Θ_0 here, and we do not tune weights for different order n-grams separately ($\Theta_1, \ldots \Theta_4$). Neither approach improved translation quality in our multi-system setting.

line search algorithm similar to golden-section search (Kiefer, 1953). We will take a closer look at this tuning procedure in Sec. 5.6.

Standard NMT models generate the translation from left to right on the target side. Recent work has shown that incorporating models which generate the target sentence in reverse order (i.e. from right to left) can improve translation quality (Sec. 3.7.5). Right-to-left models are often used to rescore n-best lists from left-to-right models. However, we could not find improvements from rescoring in our setting. Instead, we extract n-gram posteriors from the R2L model, reverse them, and use them for system combination as described above, i.e. as MBR-based scores.

4.6.2 System Setup

Data Selection

We ran language detection (Shuyo, 2010) and gentle length filtering based on the number of characters and words in a sentence on all available monolingual and parallel data in English, German, and Chinese. Due to the high level of noise in the ParaCrawl corpus and its large size compared to the rest of the English-German data we additionally filtered ParaCrawl more aggressively with the following rules:

- No words contain more than 40 characters.
- Sentences must not contain HTML tags.
- The minimum sentence length is 4 words.
- The character ratio between source and target must not exceed 1:3 or 3:1.
- Source and target sentences must be equal after stripping out non-numerical characters.
- Sentences must end with punctuation marks.

This additional filtering reduced the size of ParaCrawl from originally 36M sentences to 19M sentences after language detection, and to 11M sentences after applying the more aggressive rules.

For back-translation (Sec. 3.9) we selected 20M sentences from News Crawl 2017. We used a single Transformer (Vaswani et al., 2017) model in Tensor2Tensor's (Vaswani et al., 2018) `transformer_base` configuration for generating the synthetic source sentences. We over-sampled (Sennrich et al., 2017a) WMT data by factor 2 except in the ParaCrawl data and the UN data on Chinese-English to roughly match the size of the synthetic data. Tabs. 4.13 and 4.14 summarize the sizes of our final training corpora.

Corpus	Over-sampling	#Sentences
Common Crawl	2x	4.43M
Europarl v7	2x	3.76M
News Commentary v13	2x	0.57M
Rapid 2016	2x	2.27M
ParaCrawl	1x	11.16M
Synthetic (news-2017)	1x	20.00M
Total		**42.19M**

Table 4.13 Training data sizes for English-German and German-English after filtering.

Corpus	Over-sampling	#Sentences
CWMT - CASIA2015	2x	2.08M
CWMT - CASICT2015	2x	3.95M
CWMT - Datum2017	2x	1.93M
CWMT - NEU2017	2x	3.95M
News Commentary v13	2x	0.49M
UN v1.0	1x	14.25M
Synthetic (news-2017)	1x	20.00M
Total		**46.66M**

Table 4.14 Training data sizes for Chinese-English after filtering.

Preprocessing

We preprocess our English and German data with Moses tokenization, punctuation normalization, and truecasing. On Chinese we first used the WMT `tokenizeChinese.py`[13] script and separated segments of Chinese and Latin text from each other. Then, we removed whitespace between Chinese characters and tokenized Chinese segments with Jieba[14] and the rest with `mteval-v13a.pl`. For our neural models we apply byte-pair encoding (Sennrich et al., 2016c, BPE) with 32K merge operations. We use joint BPE vocabularies on English-German and German-English and separate source/target encodings on Chinese-English.

[13] http://www.statmt.org/wmt17/tokenizeChinese.py
[14] https://github.com/fxsjy/jieba

Architecture	en-de, de-en	zh-en
LSTM	114.2M	192.7M
SliceNet	27.5M	86.4M
Transformer	212.8M	291.4M
Relative Transformer	213.8M	292.5M

Table 4.15 Number of model parameters.

Model Hyper-parameters

We use 1024-dimensional embedding and output projection layers in all architectures. The embeddings are shared between encoder and decoder on English-German and German-English, but not on Chinese-English.

LSTM For our recurrent models we adapted the TensorFlow seq2seq tutorial code base (Luong et al., 2017) for use inside the Tensor2Tensor library (Vaswani et al., 2018).[15] We roughly followed the UEdin WMT17 submission (Sennrich et al., 2017a) and stacked four 1024-dimensional LSTM layers with layer normalization (Ba et al., 2016) and residual connections in both the decoder and bidirectional encoder. We equipped the decoder network with Bahdanau-style (Bahdanau et al., 2015) attention (`normed_bahdanau`).

SliceNet The convolutional model of Kaiser et al. (2017) called SliceNet is implemented in Tensor2Tensor. We use the standard configuration `slicenet_1` of four hidden layers with layer normalization.

Transformer We compare two Transformer variants available in Tensor2Tensor: the original Transformer (Vaswani et al., 2017) (`transformer_big` setup) and the Transformer of Shaw et al. (2018) with relative positional embeddings (`transformer_relative_big` setup). Both use 16-head dot-product attention and six 1024-dimensional encoder and decoder layers.

The number of training parameters of our neural models is summarized in Tab. 4.15.

Training

All neural models were trained with the Adam optimizer (Kingma and Ba, 2015), dropout (Srivastava et al., 2014), and label smoothing (Szegedy et al., 2016) using the Tensor2Tensor (Vaswani

[15] https://github.com/fstahlberg/tensor2tensor-usr

#Physical GPUs (g)	Delay factor (d)	#Effective GPUs (g'=gd)	Effective batch size (b'=bg')	BLEU
1	1	1	2,048	28.2
4	1	4	8,192	29.5
4	4	16	32,768	30.3
4	16	64	131,072	29.8

Table 4.16 Impact of the effective batch size on Transformer training on en-de news-test2017 after 3,276M training tokens, beam size 4.

et al., 2018) library. We decode with the average of the last 40 checkpoints (Junczys-Dowmunt et al., 2016a,b).

We make extensive use of the delayed SGD updates technique we already applied successfully to syntax-based NMT (Saunders et al., 2018). Delaying SGD updates allows to arbitrarily choose the effective batch size even on limited GPU hardware. Large batch training has received some attention in recent research (Neishi et al., 2017; Smith et al., 2017) and has been shown particularly useful for training the Transformer architecture with the Tensor2Tensor framework (Popel and Bojar, 2018). We support these findings in Tab. 4.16.[16] Our technical infrastructure[17] allows us to train on four P100 GPUs simultaneously, which limits the number of physical GPUs to $g = 4$ and the batch size[18] to $b = 2048$ due to the GPU memory. Thus, the maximum possible effective batch size without delaying SGD updates is $b' = 8192$. Training with delay factor d accumulates gradients over d batches and applies the optimizer update rule on the accumulated gradients. This allows us to scale up the effective number of GPUs to 16 and improve the BLEU score significantly (29.5 vs. 30.3). Note that training regimens are equivalent if their effective batch size is the same, ie. training on 4 physical GPUs with $d = 4$ is mathematically equivalent to training on 16 GPUs without delaying SGD updates. Tab. 4.17 lists our training setups for the neural architectures used in this work. These training hyper-parameters were chosen empirically. Particularly, we did not find improvements by increasing the number of effective GPUs for SliceNet or longer LSTM training.

We use *news-test2017* as development set on all language pairs to tune the model interpolation weights λ (Eq. 4.25) and the scaling factor for length normalization.

[16] We had to reduce the learning rate for $g' = 1$ to avoid training divergence.

[17] http://www.hpc.cam.ac.uk/

[18] We follow Vaswani et al. (2018, 2017) and specify the batch size in terms of number of source and target tokens in a batch, not the number of sentences.

Architecture	#Effective GPUs	Batch size	#SGD updates	#Training tokens
LSTM	8	4,096	45K	1,475M
SliceNet	4	2,048	800K	6,554M
R2L Transformer	16	2,048	200K	6,554M
Transformer	16	2,048	250K	8,192M
Relative Transformer	16	2,048	250K	8,192M

Table 4.17 Training setups for our neural models on all language pairs.

Decoding

We apply length normalization (Sec. 3.10.1) on German-English and Chinese-English but not on English-German. As outlined in Sec. 4.6.1 we either use full posteriors or MBR-style n-gram posteriors from our individual models. SMT n-gram scores are extracted as described in Sec. 4.5.3 from the 1000-best list of a vanilla phrase-based SMT system trained on all available data except the UN corpus and the back-translated data. We use SGNMT's (Ch. 5) `ngram` output format to extract n-gram scores from our neural models.

4.6.3 Results

On English-German and German-English *news-test2014* we compute cased BLEU scores with Moses' `multi-bleu.pl` script on tokenized output to be comparable with prior work (Kaiser et al., 2017; Vaswani et al., 2017; Wu et al., 2016b). On all other test sets we use `mteval-v13a.pl` to be comparable to the official cased WMT scores.[19]

First, we will discuss our experiments with a single architecture, i.e. single systems and ensembles of two systems with the same architecture. Tab. 4.18 compares the architectures on all test sets. PBMT as a single system is clearly inferior to all neural systems. Ensembling neural systems helps for all architectures across the board. LSTM is usually slightly better than the convolutional SliceNet, but is much slower to train and decode (cf. Tab. 4.15). Note that our LSTM 2-ensemble is on par with the best BLEU score in WMT17 (Sennrich et al., 2017a), which was also based on recurrent models. Transformer architectures outperform LSTMs and SliceNets on all test sets. The right-to-left Transformer is usually slightly worse, the Transformer with relative positioning slightly better than the standard Transformer setup.

Tab. 4.19 summarizes our system combination results with multiple architectures. Adding LSTM and SliceNet as full-posterior models to an ensemble of a Transformer and a Relative Transformer does not improve the BLEU score (rows 7 vs. 8). We see very slight improvements

[19] http://matrix.statmt.org/

Architecture	#Sys.	English-German			German-English				Chinese-English	
		test15	test16	test17	test14	test15	test16	test17	dev17	test17
PBMT	1	20.9	25.6	20.0	22.5	27.2	32.6	28.2	14.2	15.8
LSTM	1	28.8	34.6	28.0	33.8	33.3	40.7	34.8	21.8	22.7
	2	29.6	35.5	28.5	34.6	34.0	41.4	35.3	22.7	23.6
SliceNet	1	28.9	33.6	27.6	32.6	32.3	39.8	33.7	21.4	22.5
	2	29.6	34.6	28.3	33.2	32.9	40.8	34.3	21.8	23.4
R2L Trans.	1	31.5	36.3	30.2	36.5	35.5	43.5	37.2	24.5	24.9
Transformer	1	31.9	36.6	30.5	36.7	36.2	43.7	37.9	24.9	25.6
	2	31.8	37.2	31.0	36.9	36.4	44.0	38.1	26.2	26.2
Rel. Trans.	1	31.9	37.0	31.1	37.0	36.3	44.1	38.1	24.9	25.8
	2	32.3	37.7	31.2	37.2	36.5	44.1	38.4	25.1	26.4

Table 4.18 Single architecture results on all language pairs for single systems and 2-ensembles.

	System components						BLEU (test2017)		
	PBMT	LSTM*	SliceNet*	R2L Trans.	Trans.	Rel. Trans.	en-de	de-en	zh-en
1	Full						20.0	28.2	15.8
2		Full					28.5	35.3	23.6
3			Full				28.3	34.3	23.4
4				Full			30.2	37.2	24.9
5					Full		30.5	37.9	25.6
6						Full	31.1	38.1	25.8
7					Full	Full	31.3	38.2	26.4
8		Full	Full		Full	Full	31.3	38.2	26.4
9		MBR	MBR		Full	Full	31.4	38.2	26.6
10		MBR	MBR	MBR	Full	Full	31.4	38.3	26.8
11	MBR	MBR	MBR	MBR	Full	Full	31.7	38.7	27.1

Table 4.19 Model combination with ensembling and MBR. 'Full' indicates models in the "full-posterior group", 'MBR' in the 'MBR-based group' (Eq. 4.25). *: A system consisting of multiple 'Full' models is an ensemble. The LSTM and SliceNet models are already 2-ensembles by themselves.

when we use these models to extract n-gram scores instead (rows 7 vs. 9). We report further gains by using MBR-based n-gram scores from the right-to-left Transformer and the PBMT system. The improvements from adding PBMT are rather small, but we still found them surprising given that the PBMT baseline is usally more than 10 BLEU points worse than our

Direction	Test set	BLEU
English-German	news-test14	31.6
	news-test15	32.6
	news-test16	38.5
	news-test17	31.7
	news-test18	46.6
German-English	news-test14	36.8
	news-test15	36.5
	news-test16	45.1
	news-test17	38.7
	news-test18	48.0
Chinese-English	news-dev17	25.7
	news-test17	27.1
	news-test18	27.7

Table 4.20 BLEU scores of the submitted systems (row 11 in Tab. 4.19).

best single neural model. We list the performance of our submitted systems on all test sets in Tab. 4.20.

To sum up, we have described our WMT18 submission, which achieves very competitive BLEU scores on all three language pairs (English-German, German-English, and Chinese-English) and significantly higher accuracies in a variety of linguistic phenomena compared to other submissions (Macketanz et al., 2018). Our system combines three different neural architecture with a traditional PBMT system. We showed that our MBR-based scheme is effective to combine these diverse models of translation, and that adding the PBMT system to the mix of neural models still yields gains although it is much worse as stand-alone system.

4.7 Conclusion

This chapter presented a coherent line of research on SMT-NMT hybrid systems that culminated in submissions to several WMT evaluation campaigns (Stahlberg et al., 2018b, 2016a, 2019b). *Syntactically guided NMT* (Sec. 4.3) is a lattice rescoring approach that improves word-based NMT with limited vocabulary by fixing UNKs and improving NMT translation scores using Hiero. The main disadvantage of this approach is that the decoder is constrained to the SMT lattice and can only produce translations with derivations in the SMT system. Our loose combination scheme from Sec. 4.4 overcomes this limitation by combining NMT and SMT hypotheses in a formally rigorous FST-based framework via an edit distance loss. However, the

edit distance based framework can be computationally expensive as the composition of large SMT lattices with the edit distance transducer can be costly. Sec. 4.5 presented another loose combination scheme that is inspired by minimum Bayes-risk (MBR) decoding. In MBR, the SMT lattice serves as the evidence space for computing n-gram likelihoods. An unconstrained NMT decoder is biased towards likely n-grams according to the SMT lattice. MBR-based combination is computationally cheap and has been adapted for productive use (Iglesias et al., 2018). We have used a generalization of our MBR-based scheme (Sec. 4.6.1) in two successful WMT submissions (Stahlberg et al., 2018b, 2019b). Our WMT'18 English-German system has been highlighted by Macketanz et al. (2018) as it achieved significantly higher accuracies than other submissions in a variety of linguistic phenomena such as ambiguity, false friends, verb aspect/tense/mood, interrogatives, composition, etc.

The desire to be rewarded for one's creativity does not justify depriving the world in general of all or part of that creativity.

Richard Stallman

5

SGNMT – A Flexible NMT Decoding Platform

Sec. 5.4.4 on NMT search errors and model errors draws from Stahlberg and Byrne (2019b). The SGNMT tool has been described by Stahlberg et al. (2017b, 2018d), and some passages in this chapter are verbatim copies from these publications. However, this chapter goes beyond both papers and provides substantially deeper insight in the inner workings of SGNMT and its theoretical foundation, as well as descriptions of SGNMT's most recent components. This chapter is based on the SGNMT toolkit as of June 2019 (version 0.6).

5.1 Motivation

This chapter introduces SGNMT, our experimental platform for machine translation research.[1] The SGNMT decoder is used throughout this thesis and thus contains implementations of the original contributions of this thesis such as FST-constrained decoding (Secs. 4.3, 4.4, and 7.3), MBR-based NMT (Secs. 4.5 and 4.6), word reordering models and decoders (Sec. 7.2), document-level language modelling (Sec. 7.5), syntax-based NMT (Sec. 8.2), and the neural operation sequence model (Sec. 8.4). Furthermore, SGNMT is currently playing an active role

[1] Full documentation available at http://ucam-smt.github.io/sgnmt/html/.

in (1) *teaching* as SGNMT is being used for course work and student theses in the MPhil in Machine Learning and Machine Intelligence at the University of Cambridge, (2) *research* as most of the research work of the Cambridge MT group is based on SGNMT, and (3) *technology transfer* as we show how SGNMT is helping to transfer research findings from the laboratory to the industry, eg. into a product of SDL plc. SGNMT also helps us to make our research accessible and repeatable.

SGNMT provides a generic interface to neural and symbolic scoring modules (*predictors*) with left-to-right semantics such as NMT translation models, language models, translation lattices, *n*-best lists or other kinds of scores and constraints. Predictors can be combined into *predictor constellations* to form complex decoding tasks. SGNMT implements a number of search strategies (*decoders*) for traversing the space spanned by the predictors which are appropriate for different predictor constellations. Decoders in SGNMT are defined upon the predictor abstraction, which means that any search strategy is compatible with any predictor constellation. Therefore, common search procedures like beam search do not need to be reimplemented for every new model or toolkit. Adding new predictors or decoding strategies is particularly easy, making it a very efficient tool for prototyping new research ideas. The software package supports a number of well-known frameworks, including TensorFlow[2] (Abadi et al., 2016), Tensor2Tensor (Vaswani et al., 2018), OpenFST (Allauzen et al., 2007), Blocks/Theano (Bastien et al., 2012; van Merriënboer et al., 2015), and NPLM (Vaswani et al., 2013).

The strict separation of scoring module and search strategy and the decoupling of scoring modules from each other makes SGNMT a very flexible decoding tool for neural and symbolic models. Besides machine translation, SGNMT has been applied to word reordering (Hasler et al., 2017b), grammatical error correction (Stahlberg et al., 2019a), and music composition (Tomczak, 2016), often within competitive evaluation campaigns (Stahlberg and Byrne, 2019a; Stahlberg et al., 2018b, 2016a, 2019b; Yuan et al., 2019). Hasler et al. (2017b) demonstrated the versatility of SGNMT for word reordering (Sec. 7.2) by combining five very different models (RNN LM, feedforward NPLM, Kneser-Ney LM, bag-to-seq model, seq-to-seq model) and a bag-of-words constraint using predictors.

5.2 NMT Toolkits

The rate of innovation in machine translation (MT) has gathered impressive momentum over the recent years. The discovery and maturation of the NMT paradigm (Ch. 3) has led to steady and substantial improvements of translation performance. Fig. 5.1 shows that this progress is often driven by significant changes in the network architecture. This volatility poses major

[2]SGNMT relies on the TensorFlow fork available at https://github.com/ehasler/tensorflow

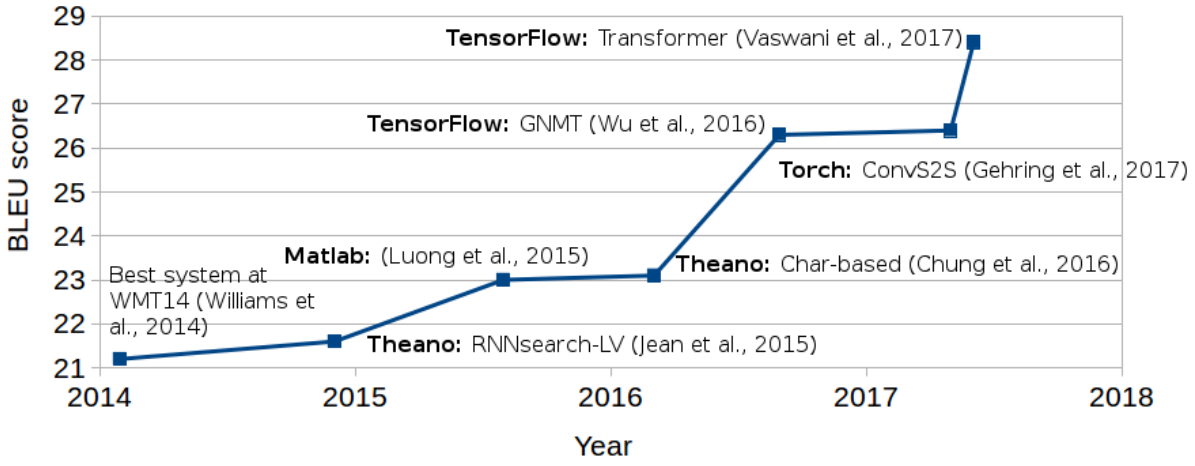


Fig. 5.1 Best systems on the English-German WMT *news-test2014* test set over the years (BLEU script: Moses' `multi-bleu.pl`), indicating the changes in models and experimental infrastructure.

challenges in MT-related research, teaching, and industry. Researchers potentially spend a lot of time implementing to keep their setups up-to-date with the latest models, teaching needs to identify suitable material in a changing environment, and the industry faces demanding speed requirements on its deployment processes.

Another practical challenge many researchers are struggling with is the large number of available NMT tools. Tab. 5.1 lists NMT-related software projects that are still actively being developed as of Februray 2019. Committing to one particular NMT tool bears the risk of being outdated soon, as keeping up with the pace of research is especially costly for NMT software developers.

Our SGNMT platform is an attempt to mediate the effects of the rapid progress in MT and the diversity of available NMT software. We can think of an SGNMT *predictor* as an interface to a particular neural model or NMT tool. The interface also allows implementing constraints such as lattice or n-best list rescoring, and symbolic models such as n-gram language models or counting models as predictors. Our software architecture is designed to facilitate the implementation of new predictors. Therefore, SGNMT can be extended to a new model or tool with very limited coding effort because rather than reimplementing models it is often enough to access APIs within an adapter predictor.[3] Software packages which are not written in Python can be exposed in SGNMT if they have a Python interface.[4] Once a new predictor is implemented, it can be directly combined with all other predictors which are already available in SGNMT.

[3]Making all models of the T2T library (Vaswani et al., 2018) available to SGNMT took less than 200 lines of code.

[4]For example, the neural language modeling software NPLM (Vaswani et al., 2013) is written in C++, but can be accessed in SGNMT via its Python interface.

Name	Citation	Framework	GitHub Stars
Tensor2Tensor	Vaswani et al. (2018)	TensorFlow	
TensorFlow/NMT	-	TensorFlow	
Fairseq	Ott et al. (2019)	PyTorch	
OpenNMT-py	Klein et al. (2017)	Lua, (Py)Torch, TF.	
Sockeye	Hieber et al. (2017)	MXNet	
OpenSeq2Seq	Kuchaiev et al. (2018)	TensorFlow	
Nematus	Sennrich et al. (2017b)	TensorFlow, Theano	
PyTorch/Translate	-	PyTorch	
Marian	Junczys-Dowmunt et al. (2016a)	C++	
NMT-Keras	Álvaro Peris and Casacuberta (2018)	TensorFlow, Theano	
Neural Monkey	Helcl and Libovický (2017)	TensorFlow	
THUMT	Zhang et al. (2017c)	TensorFlow, Theano	
Eske/Seq2Seq	-	TensorFlow	
XNMT	Neubig et al. (2018)	DyNet	
NJUNMT	-	PyTorch, TensorFlow	
Transformer-DyNet	-	DyNet	
SGNMT	Stahlberg et al. (2017b, 2018d)	TensorFlow, Theano	
CythonMT	Wang et al. (2018h)	C++	
Neutron	Xu and Liu (2019)	PyTorch	

Table 5.1 NMT toolkits that have been updated in the past year (as of February 2019). GitHub stars indicate the popularity of tools on GitHub.

This does not only speed up the transition to a new NMT toolkit, it also allows the combination of different NMT implementations, eg. ensembling a Theano-based NMT model (Bastien et al., 2012; van Merriënboer et al., 2015) with a TensorFlow-based Tensor2Tensor (Vaswani et al., 2018) model.

5.3 Predictors

SGNMT consequently emphasizes flexibility and extensibility by providing a common interface to a wide range of constraints or models used in MT research via predictors. Our platform aims to minimize the effort required for implementation; decoding speed is secondary as optimized code for production systems can be produced once an idea has been proven successful in the SGNMT framework. In SGNMT, scores are assigned to partial hypotheses via one or many predictors. One predictor usually has a single responsibility as it represents a single model or type of constraint. Predictors need to implement the following methods:

Predictor	**Predictor state**	`initialize(·)`	`predict_next()`	`consume(token)`
NMT (recurrent)	State vector in the GRU or LSTM layer of the decoder network and current context vector.	Run encoder network to compute annotations.	Forward pass through the decoder to compute the posterior given the current decoder GRU or LSTM state and the context vector.	Feed back `token` to the NMT network and update the decoder state and the context vector.
FST	ID of the current node in the FST.	Load FST from the file system, set the predictor state to the FST start node.	Explore all outgoing edges of the current node and use arc weights as scores.	Traverse the outgoing edge from the current node labelled with `token` and update the predictor state to the target node.
n-gram	Current n-gram history	Set the current n-gram history to the begin-of-sentence symbol.	Return the LM scores for the current n-gram history.	Add `token` to the current n-gram history.
Word count	None	Empty	Return a cost of 1 for all tokens except `</s>`.	Empty

Table 5.2 Predictor operations for the NMT, FST, n-gram LM, and counting modules.

- `initialize(src_sentence)` Initialize the predictor state using the source sentence.
- `get_state()` Get the internal predictor state.
- `set_state(state)` Set the internal predictor state.
- `predict_next()` Given the internal predictor state, produce the scores of target tokens for the next position.
- `consume(token)` Update the internal predictor state by adding `token` to the current history.

The structure of the predictor state and the implementations of these methods differ substantially between predictors. Tab. 5.2 summarizes the semantics of this interface for three very common predictors: the neural machine translation (NMT) predictor, the (deterministic) finite state transducer (FST) predictor for lattice rescoring, and the n-gram predictor for applying n-gram language models. We also included two examples (word count and UNK count) which do not have a natural left-to-right semantic but can still be represented as predictors. Tab. 5.3

Predictor	Description
t2t	Predictor for Tensor2Tensor (Vaswani et al., 2018) models.
fertt2t, segt2t	Variants of the t2t predictor for fertility modeling or document-level models as in Sec. 7.5.
nmt	Recurrent NMT following Bahdanau et al. (2015). Supports Blocks/Theano (Bastien et al., 2012; van Merriënboer et al., 2015) and TensorFlow (Abadi et al., 2016).
(n)fst	Predictor for rescoring (non)deterministic lattices (Stahlberg et al., 2016b).
rtn	Rescoring recurrent transition networks as created by HiFST (Allauzen et al., 2014) with late expansion.
(lex)nizza	Support for the neural alignment models.
srilm, kenlm, nplm	n-gram language model using the SRILM (Stolcke, 2002), KenLM (Heafield, 2011; Heafield et al., 2013), or NPLM (Vaswani et al., 2013) toolkits.
rnnlm	Integrates RNN language models with TensorFlow as described by Zaremba et al. (2014).
forced(lst)	Forced decoding with a single reference or n-best list.
bow	Restricts the search space to a bag of words as described in Sec. 7.2.
lrhiero	Experimental implementation of left-to-right Hiero (Siahbani et al., 2013) for small grammars.
wc	Number of words feature.
unkc	Applies a Poisson model for the number of UNKs in the output.
ngramc	Integrates external n-gram posteriors, e.g. for MBR-based NMT (Sec. 4.5).
(forced)osm	Support for the neural operation sequence model presented in Sec. 8.4.
bracket	Enforce well-formed bracket expressions, e.g. for syntax-based NMT (Sec. 8.2)
(ext)length	Target sentence length model using simple source sentence features or an external length distribution.

Table 5.3 Currently implemented predictors as of June 2019 (SGNMT version 0.6).

```
class NMTPredictor(Predictor):
  def initialize(src_sentence):
    enc_states = enc_computation_graph(
        src_sentence)
    dec_input = [BOS]
  def predict_next():
    scores, dec_state = \
        dec_computation_graph(
            dec_input, enc_states)
    return scores
  def consume(word):
    dec_input = word
  def get_state():
    return dec_state, dec_input
  def set_state(state):
    dec_state, dec_input = state
```

(a) The *nmt* predictor for recurrent models

```
class FSTPredictor(Predictor):
  def initialize(src_sentence):
    Load FST file
    cur_node = start_node

  def predict_next():
    return outgoing_arcs(cur_node)

  def consume(word):
    cur_node = cur_node.arcs[word]

  def get_state():
    return cur_node

  def set_state(state):
    cur_node = state
```

(b) The *fst* predictor

Fig. 5.2 Pseudo-code predictor implementations

```
predictors: fst, t2t, t2t
src_test: ./data/bpes/test.bpe.ids.ja  # Path to the input sentences
fst_path: ./lattices.test/%d.fst  # Path to lattices

# T2T model specification (Transformer base)
pred_src_vocab_size: 35786
pred_trg_vocab_size: 32946
t2t_problem: translate_jaen_kyoto32k
t2t_model: transformer
t2t_hparams_set: transformer_base
t2t_checkpoint_dir: ./t2t_train/transformer/  # Path to the first T2T model
t2t_checkpoint_dir2: ./t2t_train/transformer2/  # Path to the second model
```

Fig. 5.3 Example SGNMT configuration file specifying ensembled lattice rescoring with two T2T models.

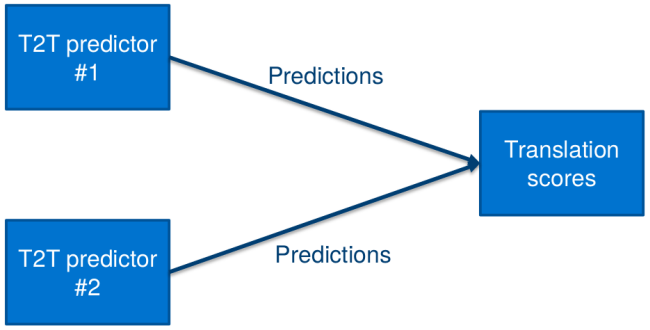
lists all predictors which are currently implemented. Pseudo-code for two example predictors is listed in Fig. 5.2.

SGNMT can be configured via command-line arguments and configuration files (.ini format). An example of a complete SGNMT configuration file for lattice rescoring with an ensemble is given in Fig. 5.3.

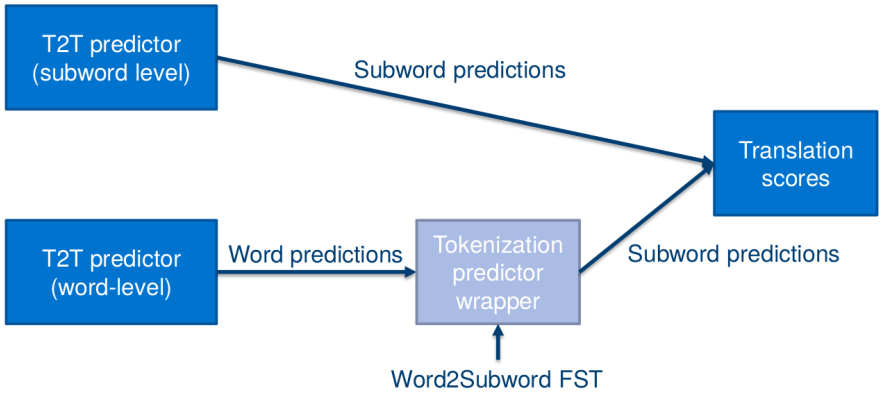
5.3.1 Example Predictor Constellations

SGNMT allows combining any number of predictors and even multiple instances of the same predictor type. In case of multiple predictors we combine the predictor scores in a linear model. The following list illustrates that various interesting decoding tasks can be formulated as predictor combinations.

- `nmt`: A single NMT predictor represents pure NMT decoding (Sec. 3.7).
- `nmt,nmt,nmt`: Using multiple NMT predictors is a natural way to represent ensemble decoding (Sec. 3.7.4) in our framework.
- `fst,nmt`: NMT decoding constrained to an FST. This can be used for neural lattice rescoring (Stahlberg et al., 2016b) or other kinds of constraints, for example in the context of source side simplification in MT (Hasler et al., 2017a) or chord progressions in 'Bach' chorales (Tomczak, 2016). The *fst* predictor can also be used to restrict the output of character-based or subword-unit-based NMT to a large word-level vocabulary encoded as FSA.



(a) Normal ensembling of two T2T models.



(b) Ensembling of a word-level and a subword-level model.

Fig. 5.4 Multi-tokenization ensembles with the *fsttok* predictor wrapper.

```
predictors: t2t,fsttok_t2t
fsttok_path: word2bpe.fst
t2t_checkpoint_dir: ./t2t_train/bpe_transformer/
t2t_checkpoint_dir2: ./t2t_train/word_transformer/
src_test: ./data/bpes/test.bpe.ids.ja  # Path to the input sentences
# T2T model specification
...
```

Fig. 5.5 SGNMT .ini-file for Fig. 5.4b.

- `nmt,rnnlm,srilm,nplm`: Combining NMT with three kinds of language models: An RNNLM (Zaremba et al., 2014), a Kneser-Ney *n*-gram LM (Heafield et al., 2013; Stolcke, 2002), and a feedforward neural network LM (Vaswani et al., 2013).
- `nmt,ngramc,wc`: MBR-based NMT as presented in Sec. 4.5 with *n*-gram posteriors extracted from an SMT lattice (`ngramc`) and a simple word penalty (`wc`).

5.3.2 Predictor Wrappers

Predictors can be masked by one or many predictor *wrappers*. Wrappers change the behavior of a predictor, for example by manipulating the data which is fed into it or by modifying

Predictor	Description
idxmap	1:1 mapping between word IDs.
altsrc	This wrapper loads source sentences from an alternative file.
rank	Does not use the scores of the wrapped predictor directly but the rank in the scores table.
glue	Masks sentence-level predictors when SGNMT runs on the document level.
ngramize	Extracts *n*-gram posteriors from a predictor without feedback loop.
skipvocab	Uses internal beam search to skip a subset of the predictor vocabulary.
maskvocab	Hides a subset of the SGNMT vocabulary from the wrapped predictor.
fsttok	Uses an FST to transduce SGNMT tokens to predictor tokens.
word2char	Wraps word-level predictors when SGNMT is running on the character level.
parse	Internal beam search over a representation which contains some pre-defined non-terminal ids, which should not appear in the output. Used for syntax-based multi-representation ensembles (Sec. 8.2)

Table 5.4 Currently implemented predictor wrappers as of June 2019 (SGNMT version 0.6).

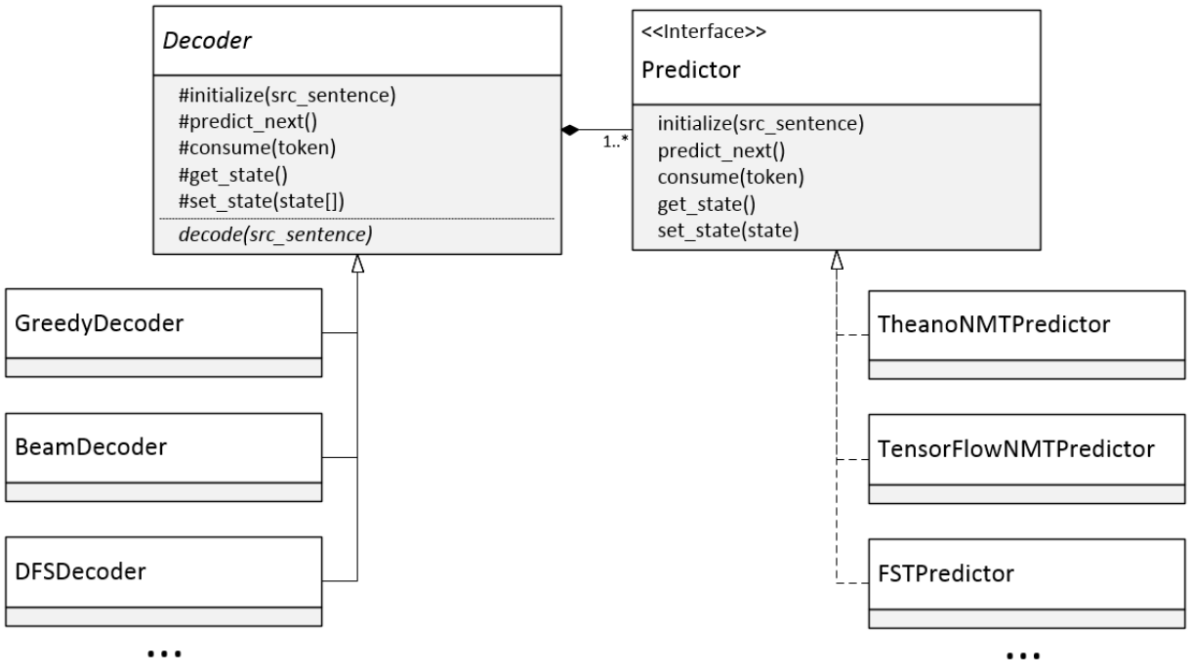


Fig. 5.6 Reduced Unified Modeling Language (UML) class diagram.

the predictions from it. Wrappers are transparent to the rest of the system. An example of a predictor wrapper is *fsttok* which allows ensembling with models at multiple tokenization levels (Stahlberg et al., 2018d). The conversion between the tokenization schemes of different predictors is defined with FSTs. This makes it possible to decode by combining scores from both a subword-unit (BPE) based NMT (Sennrich et al., 2016c) and a word-based NMT model with character-based NMT, masking the BPE-based and word-based NMT predictors with FSTs which transduce character sequences to BPE or word sequences. Fig. 5.4 illustrates ensembling of a subword-unit-based and a word-based T2T model, Fig. 5.5 lists the corresponding .ini-file. Joint decoding with different tokenization schemes has the potential of combining the benefits of the different schemes: character- and BPE-based models are able to address rare words, but word-based NMT can model long-range dependencies more efficiently.

Tab. 5.4 lists all available predictor wrappers as of June 2019 (SGNMT version 0.6).

5.4 Decoders

In Sec. 3.7.3 we have defined greedy and beam search for a specific NMT model, the RNNsearch model of Bahdanau et al. (2015). In this section, we present SGNMT *decoders* as a more general framework to view decoding. *Decoders* are algorithms to search for the highest scoring hypothesis. The list of predictors determines how (partial) hypotheses are scored by

Algorithm 4 Greedy(src_sen)

```
initialize(src_sen)
h ← ⟨<s>⟩
repeat
    P ←predict_next()
    (t, c) ← arg max_{(t',c')∈P} c'
    h ← h · t
    consume(t)
until t = </s>
return h
```

Algorithm 5 Beam(n, src_sen)

```
initialize(src_sen)
H ← {(⟨<s>⟩, 0.0, get_state())}
repeat
    H_next ← ∅
    for all (h, c, s) ∈ H do
        set_state(s)
        P ←predict_next()
        H_next ← H_next ∪ ⋃_{(t',c')∈P} (h · t', c + c', s)
    end for
    H ← ∅
    for all (h, c, s) ∈ n-best(H_next) do
        set_state(s)
        consume(h_{|h|})
        H ← H ∪ {(h, c, get_state())}
    end for
until Best hypothesis in H ends with </s>
return Best hypothesis in H
```

implementing the methods `initialize(·)`, `get_state()`, `set_state(·)`, `predict_next()`, and `consume(·)`. The *Decoder* class implements versions of these methods which apply to all predictors in the list. Decoder implementations access the predictors exclusively via these methods, and are thus independent of the predictor constellation. This design decision is crucial to ensure that decoders are compatible to any model, toolkit, constraint, etc. that implements the `Predictor` interface. Fig. 5.6 illustrates the relation between decoders and predictors.

Many popular search strategies can be described via this interface. Algs. 4 and 5 show how greedy and beam search can be defined that way.[5] Tab. 5.5 lists all decoders currently available in SGNMT v0.6 (as of June 2019).

[5]Formally, `predict_next()` in Algs. 4 and 5 returns pairs of tokens and their costs. String concatenation is denoted with $\cdot$.

Decoder	Description
greedy	Greedy decoding.
beam	Beam search as described by Bahdanau et al. (2015).
dfs	Depth-first search. Efficiently enumerates the complete search space.
restarting	Depth-first search with better pruning behavior.
astar	A* search (Russell et al., 2003). The heuristic function can be defined via predictors.
sepbeam	Associates hypotheses in the beam with only one predictor. Efficiently approximates system-level combination.
syncbeam	Beam search which compares hypotheses after consuming a special synchronization symbol rather than after each iteration.
bucket	Multiple beam search passes with small beam size. Can have better pruning behaviour than standard beam search.
syntaxbeam	Beam search which ensures diversity amongst terminal symbol histories.
mbrbeam	Diversity encouraging beam search which maximizes the expected BLEU.
multisegbeam	Beam search with multiple segmentations.
flip	This decoder works only for bag problems. It traverses the search space by switching two words in the hypothesis.
bow	Restarting decoder optimized for bag-of-words problems.
bigramgreedy	Works best for bag problems. Collects bigram statistics and constructs hypos to score by greedily selecting high scoring bigrams.
vanilla	Fast beam search decoder for (ensembled) NMT. This implementation is similar to the decoder in Blocks (van Merriënboer et al., 2015) but can only be used for NMT as it bypasses the predictor framework.

Table 5.5 Currently implemented decoding strategies as of June 2019 (SGNMT version 0.6).

5.4.1 Neural Decoding as Shortest Path Search

We have introduced SGNMT's *decoders* as a way to define search procedures for (combinations of) predictors. The software architecture of SGNMT is motivated by well-known software engineering principles such as decoupling ("high cohesion – loose coupling") (Bourque et al., 2014). In this section, we set a rigorous foundation of the SGNMT framework by using formalisms from finite state transducers (FSTs). In particular, we argue that SGNMT decoding is very similar to navigating through a finite state machine, with the only difference that the state space my not be finite.[6]

[6]Most predictors induce infinite search spaces. For example, the NMT search space is not finite.

Algorithm 6 GreedySearch

$\mathbf{y} \leftarrow \varepsilon$
$v \leftarrow s$
while $v \notin F$ **do**
  $\hat{e} \leftarrow \arg\min_{e \in E_v} \omega[e]$
  $v \leftarrow n[\hat{e}]$
  $\mathbf{y}$.append$(o[\hat{e}])$
end while
return $\mathbf{y}$

Algorithm 7 BeamSearch$(n \in \mathbb{N})$

$\mathscr{H}_{cur} = E_s$
repeat
  $\mathscr{H}_{next} \leftarrow \emptyset$
  for all $p \in \mathscr{H}_{cur}$ **do**
    $v \leftarrow n[\pi_{|p|}(p)]$ {v is the last state in path p}
    if $v \in F$ **then**
      $\mathscr{H}_{next} \leftarrow \mathscr{H}_{next} \cup \{p\}$ {Complete paths are not extended}
    else
      $\mathscr{H}_{next} \leftarrow \mathscr{H}_{next} \cup \bigcup_{e \in E_v} p \cdot e$ {Add all possible continuations}
    end if
  end for
  $\mathscr{H}_{cur} \leftarrow \{p \in \mathscr{H}_{next} : |\{p' \in \mathscr{H}_{next} : \omega[p'] < \omega[p]\}| < n\}$ {Select n-best}
  $\hat{p} \leftarrow \arg\min_{p \in \mathscr{H}_{cur}} \omega[p]$
until $n[\pi_{|\hat{p}|}(\hat{p})] \in F$
return $o[\hat{p}]$

We follow our notation introduced in Sec. 2.6 based on Mohri (2003) and consider the SGNMT search space as a (potentially infinite) state transducer $T = (V, s, F, \Sigma_{trg}, \Sigma_{trg}, E)$. Similarly to Eq. 4.4, if a path p to a state $v \in V$ accepts the translation prefix y_1^{j-1}, the weight on an outgoing arc $e \in E_v$ from that state with symbol $y = o[e]$ corresponds to the partial predictor score. For example, for pure NMT decoding, this is the negative log of the conditional probability

$$-\log P(y_j = y | y_1^{j-1}, \mathbf{x}) = \omega[e]. \tag{5.1}$$

Additionally, in this section we assume that the initial state is not a final state ($s \notin F$). Let $c \in V$ denote the current state. The methods of SGNMT's Decoder interface can be understood as navigation instructions through T:

- `initialize(·)` Set the current state c to the start state $s \in V$.
- `get_state()` Get the current state c.

Algorithm 8 DepthFirstSearch

1: $\mathscr{H} \leftarrow \bigcup_{e \in E_s} \text{DFSVisit}(e)$
2: **return** $o[\arg\min_{p \in \mathscr{H}} \omega[p]]$

Algorithm 9 DFSVisit($p \in E^+$)

1: $r \leftarrow n[\pi_{|p|}(p)]$
2: **if** $r \in F$ **then**
3: **return** p {Return if last state in path p is final}
4: **end if**
5: $\mathscr{H} \leftarrow \bigcup_{e \in E_r} \text{DFSVisit}(p \cdot e)$
6: **return** $\arg\min_{p \in \mathscr{H}} \omega[p]$

- `set_state(state)` Set the current state c to `state`.
- `predict_next()` For each outgoing arc $e \in E_c$, return the pair $(\omega[e], o[e])$ of the next symbol and its score.
- `consume(token)` Look up the arc $e \in E_c$ that is labeled with `token` ($o[e] =$ `token`). Update the current state to the next state along e: $c = n[e]$.

Note that it would be very costly if not impossible to construct T explicitly, and that this section is merely intended to demonstrate the theoretical link between SGNMT decoding and FSTs.

A reformulation of greedy and beam search as shortest path search in an (in)finite state machine is shown in Algs. 6 and 7. We will take a closer look at two more decoders implemented in SGNMT from the FST perspective: depth-first search and A*. The algorithm descriptions in this section follow Cormen (2009) and Russell et al. (2003).

Depth-first Search

We have already pointed out in Sec. 4.3.2 that even constrained NMT has a tree-structured search space due to the unbounded NMT history length. Searching on trees often greatly simplifies search algorithms as we do not have to keep track of already visited nodes. The depth-first search (DFS) strategy searches " 'deeper' in the graph whenever possible" (Cormen, 2009, Sec. 22.3). In contrast to time synchronous search like beam or breadth-first search, it explores a branch of the search tree exhaustively before "backtracking" into another branch. Therefore, it always explores the state in the current frontier with maximum path length to the initial state s (Russell et al., 2003, Sec. 3.4.3). Alg. 9 is a recursive DFS implementation for tree structures.

Algorithm 10 A*

$\mathscr{H} = E_s$
while $\mathscr{H} \neq \emptyset$ **do**
  $p \leftarrow \arg\min_{p \in \mathscr{H}} \omega[p] \otimes h(n[\pi_{|p|}(p)])$ {Get path p with minimum estimated cost.}
  $v \leftarrow n[\pi_{|p|}(p)]$
  if $v \in F$ **then**
    return $o[p]$
  end if
  $\mathscr{H} \leftarrow \mathscr{H} \setminus \{p\} \cup \bigcup_{e \in E_v} p \cdot e$ {Remove p from $\mathscr{H}$ and add all possible continuations.}
end while
return *failure*

A* Search

A* is an informed search strategy (Russell et al., 2003, Sec. 3.5), i.e. it selects the next state to expand based on additional knowledge about the search state. This additional knowledge is encapsulated in a heuristic function $h : V \to \mathbb{K}$. For a state $v \in V$, $h(v)$ is the estimated cost of the cheapest path from v to a final state. A* expands the active state v which minimizes the following term (Russell et al., 2003, Sec. 3.5.2):

$$f(v) = g(v) \otimes h(v) \tag{5.2}$$

where $g(v)$ is the cost of the path p from the initial node s to v (e.g. $\omega[p]$). Therefore, $f(v)$ is the estimated cost of the cheapest path in $\mathscr{P}(T)$ (set of complete paths as defined in Eq. 2.9) through v. Alg. 10 shows the pseudo code for A*.

A* is guaranteed to find the optimal path if the heuristic $h(\cdot)$ is *admissible*[7]. That means that $h(\cdot)$ never overestimates the true future cost. SGNMT offers different ways to define heuristics. First, predictors can implement their own heuristic functions. For example, the *fst* predictor can use the shortest distance from v to a final state in the lattice. This is very fast, but often not very informative as the *fst* predictor is usually paired with a neural model which receives a much higher predictor weight (e.g. λ_{Hiero} is usually much smaller than λ_{NMT} in Eq. 4.2). Alternatively, we can do greedy decoding from v. This is much more expensive but more accurate.

Note that A* is a generalization of breadth-first and depth-first search. Let p be the path from initial state s to a state $v \in V$ and $h \equiv \bar{1}$. If we set $g(v) = |p|$ we recover breadth-first search, setting $g(v) = -|p|$ results in depth-first search.

[7] Russell et al. (2003) also require $h(\cdot)$ to be *consistent* but we do not need this condition as our search space is a tree.

5.4.2 NMT Batch Decoding

The flexibility of the predictor framework comes with degradation in decoding time. SGNMT provides two ways of speeding up pure NMT decoding, especially on the GPU. The *vanilla* decoding strategy exposes the beam search implementation in Blocks (van Merriënboer et al., 2015) which processes all active hypotheses in the beam in parallel. We also implemented a beam decoder version which decodes multiple sentences at once (batch decoding) rather than in a sequential order. Batch decoding is potentially more efficient since larger batches can make better use of GPU parallelism. The key concepts of our batch decoder implementation are:

- We use a scheduler running on a separate CPU thread to construct large batches of computation (GPU jobs) from multiple sentences and feeding them to the *jobs* queue.
- The GPU is operated by a single thread which communicates with the CPU scheduler thread via queues containing jobs. This thread is only responsible for retrieving jobs in the *jobs* queue, computing them, and putting them in the *jobs_results* queue, minimizing the down-time of GPU computation.
- Yet another CPU thread is responsible for processing the results computed on the GPU in the *job_results* queue, e.g. by getting the n-best words from the posteriors. Processed jobs are sent back to the CPU scheduler where they are reassembled into new jobs.

This decoder is able to translate the WMT English-French test sets *news-test2012* to *news-test2014* on a Titan X GPU with 911.6 words per second with the word-based NMT model described in Stahlberg et al. (2017b).[8] This decoding speed seems to be slightly faster than sequential decoding with high-performance NMT decoders like Marian-NMT (Junczys-Dowmunt et al., 2016a) with reported decoding speeds of 865 words per second.[9] However, batch decoding with Marian-NMT is much faster reaching over 4,500 words per second.[10] We think that these differences are mainly due to the limited multithreading support and performance in Python especially when using external libraries as opposed to the highly optimized C++ code in Marian-NMT. We did not push for even faster decoding as speed is not a major design goal of SGNMT. Note that batch decoding bypasses the predictor framework and can only be used for pure NMT decoding.

[8] Theano 0.9.0, cuDNN 5.1, Cuda 8 with CNMeM, Intel® Core i7-6700 CPU

[9] Note that the comparability is rather limited since even though we use the same beam size (5) and vocabulary sizes (30k), we use (a) a slightly slower GPU (Titan X vs. GTX 1080), (b) a different training and test set, (c) a slightly different network architecture, and (d) words rather than subword units.

[10] https://marian-nmt.github.io/features/

5.4.3 Exact Inference in NMT

Recall Eq. 3.1 that states that NMT assigns the probability $P(\mathbf{y}|\mathbf{x})$ of a translation $\mathbf{y} = y_1^J \in \Sigma_{\text{trg}}^J$ of length J for a source sentence $\mathbf{x} \in \Sigma_{\text{src}}^I$ of length I via a left-to-right factorization using the chain rule:

$$\log P(\mathbf{y}|\mathbf{x}) = \sum_{j=1}^{J} \log P(y_j|y_1^{j-1}, \mathbf{x}). \tag{5.3}$$

The task of finding the most likely translation $\hat{\mathbf{y}} \in \Sigma_{\text{trg}}^*$ for a given source sentence $\mathbf{x}$ is known as the *decoding* or *inference* problem. We argued in Sec. 3.7 that the NMT search space is far too big for exhaustive enumeration. The size of the NMT search space is perhaps the main reason why – besides some preliminary studies (Niehues et al., 2017; Ott et al., 2018a; Stahlberg et al., 2018d) – analyzing search errors in NMT has received only limited attention. To the best of our knowledge, none of the previous studies were able to quantify the number of search errors in unconstrained NMT due to the lack of an exact inference scheme that – although too slow for practical MT – guarantees to find the global best model score for analysis purposes.

In (Stahlberg and Byrne, 2019b), we proposed such an exact decoding algorithm for NMT that exploits the monotonicity of NMT scores: Since the conditional log-probabilities in Eq. 5.3 are always negative, partial hypotheses can be safely discarded once their score drops below the log-probability of any *complete* hypothesis. Using our exact inference scheme we show that beam search does not find the global best model score for more than half of the sentences. However, these *search* errors, paradoxically, often prevent the decoder from suffering from a frequent but very serious *model* error in NMT, namely that the empty hypothesis often gets the global best model score. Our findings suggest a reassessment of the amount of model and search errors in NMT, and we hope that they will spark new efforts in improving NMT modeling capabilities, especially in terms of adequacy.

Exact Inference for Neural Models Beam search is the ubiquitous decoding algorithm for NMT (Sec. 3.7), but it is prone to search errors as the number of active hypotheses is limited by the beam size. In particular, beam search never compares partial hypotheses of different lengths with each other. As we will see in later sections, this is one of the main sources of search errors. However, in many cases, the model score found by beam search is a reasonable approximation to the global best model score. Let γ be the model score found by beam search, which is a lower bound on the global best model score: $\gamma \leq \log P(\hat{\mathbf{y}}|\mathbf{x})$. Furthermore, since the conditionals $\log P(y_j|y_1^{j-1}, \mathbf{x})$ in Eq. 5.3 are log-probabilities and thus non-positive, expanding

Algorithm 11 ExactSearch($\mathbf{x}, \mathbf{y}, p \in \mathbb{R}, \gamma \in \mathbb{R}$)

Require: $\mathbf{x}$: Source sentence
$\mathbf{y}$: Translation prefix (default: ε)
p: $\log P(\mathbf{y}|\mathbf{x})$ (default: 0.0)
γ: Lower bound

1: **if** $y_{|\mathbf{y}|} = <\!/\mathtt{s}\!>$ **then**
2: **return** $(\mathbf{y}, p)$ {Trigger γ update}
3: **end if**
4: $\tilde{\mathbf{y}} \leftarrow \perp$ {Initialize $\tilde{\mathbf{y}}$ with dummy value}
5: **for all** $w \in \Sigma_{\text{trg}}$ **do**
6: $p' \leftarrow p + \log P(w|\mathbf{x}, \mathbf{y})$
7: **if** $p' \geq \gamma$ **then**
8: $(\mathbf{y}', \gamma') \leftarrow \text{DFS}(\mathbf{x}, \mathbf{y} \cdot w, p', \gamma)$
9: **if** $\gamma' > \gamma$ **then**
10: $(\tilde{\mathbf{y}}, \gamma) \leftarrow (\mathbf{y}', \gamma')$
11: **end if**
12: **end if**
13: **end for**
14: **return** $(\tilde{\mathbf{y}}, \gamma)$

a partial hypothesis is guaranteed to result in a lower model score, i.e.:[11]

$$\forall j \in [2, J] : \log P(y_1^{j-1}|\mathbf{x}) > \log P(y_1^j|\mathbf{x}). \tag{5.4}$$

Consequently, when we are interested in the global best hypothesis $\hat{\mathbf{y}}$, we only need to consider partial hypotheses with scores greater than γ. In our exact decoding scheme we traverse the NMT search space in a depth-first order, but cut off branches along which the accumulated model score falls below γ. During depth-first search (DFS), we update γ when we find a better complete hypothesis. Alg. 11 specifies the DFS algorithm formally. An important detail is that elements in Σ_{trg} are ordered such that the loop in line 5 considers the $<\!/\mathtt{s}\!>$ token first. This often updates γ early on and leads to better pruning in subsequent recursive calls.[12]

Chen et al. (2018c) showed that the consistent best string problem for RNNs is decidable. We provide here an alternative DFS algorithm that relies on the monotonic nature of model scores rather than consistency, and that often converges in practice. If this exact search terminates we know for sure that the resulting hypothesis is the global best. But, and in particular for inconsistent RNNs, this procedure is not guaranteed to terminate.

[11] Equality in Eq. 5.4 is impossible since probabilities are modeled by the neural model via a softmax function which never predicts a probability of *exactly* 1.

[12] Note that the order in which the for-loop in line 5 of Alg. 11 iterates over Σ_{trg} may be important for efficiency but does not affect the correctness of the algorithm.

Search	BLEU	Ratio	#Search errors	#Empty
Greedy	29.3	1.02	73.6%	0.0%
Beam-10	30.3	1.00	57.7%	0.0%
Exact	2.1	0.06	0.0%	51.8%

Table 5.6 NMT with exact inference. In the absence of search errors, NMT often prefers the empty translation, causing a dramatic drop in length ratio and BLEU.

Exact inference under length constraints Our admissible pruning criterion based on γ relies on the fact that the model score of a (partial) hypothesis is always lower than the score of any of its translation prefixes. While this monotonicity condition is true for vanilla NMT (Eq. 5.4), it does not hold for methods like length normalization (Boulanger-Lewandowski et al., 2013; Jean et al., 2015b; Wu et al., 2016b) or word rewards (He et al., 2016d): Length normalization gives an advantage to longer hypotheses by dividing the score by the sentence length, while a word reward directly violates monotonicity as it rewards each word with a positive value. We will show later how our exact search can be extended to handle arbitrary length models (Huang et al., 2017a; Murray and Chiang, 2018; Yang et al., 2018b) by introducing length dependent lower bounds γ_k and report initial findings on exact search under length normalization. However, despite being of practical use, methods like length normalization and word penalties are rather heuristic as they do not have any justification from a probabilistic perspective. They also do not generalize well as (without retuning) they often work only for a specific beam size. It would be much more desirable to fix the length bias in the NMT model itself.

5.4.4 Using Exact Inference to Characterize NMT Search Errors and Model Errors

Results without Length Constraints We conduct all our experiments in this section on the entire English-German WMT `news-test2015` test set (2,169 sentences) with a Transformer base (Vaswani et al., 2017) model trained with Tensor2Tensor (Vaswani et al., 2018) on parallel WMT18 data excluding ParaCrawl. Our pre-processing is as described by Stahlberg et al. (2018b) and includes joint subword segmentation using byte pair encoding (Sennrich et al., 2016c) with 32K merges. We report cased BLEU scores.[13] An open-source implementation of our exact inference scheme is available in the SGNMT decoder (Stahlberg et al., 2017b, 2018d).[14]

[13] Comparable with http://matrix.statmt.org/

[14] `simpledfs` decoding strategy in SGNMT.

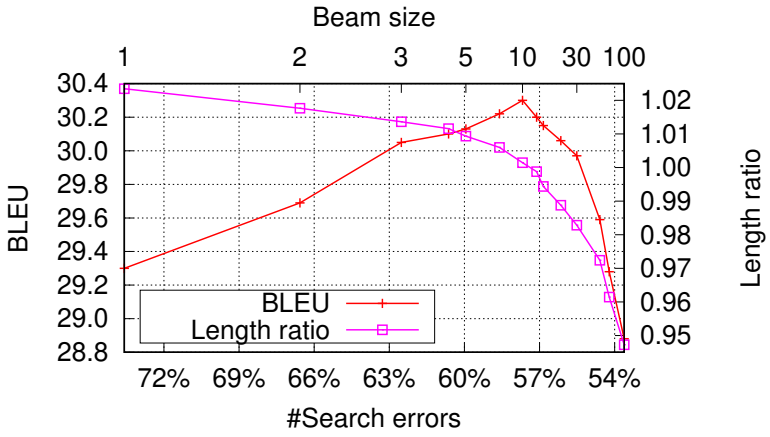


Fig. 5.7 BLEU over the percentage of search errors. Large beam sizes yield fewer search errors but the BLEU score suffers from a length ratio below 1.

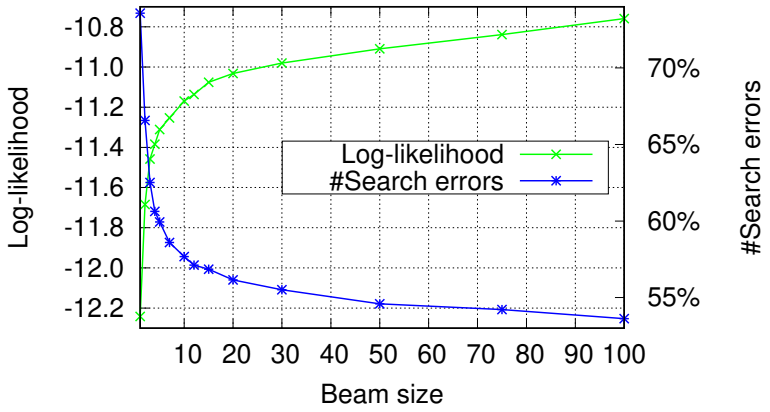


Fig. 5.8 Even large beam sizes produce a large number of search errors.

Our main result is shown in Tab. 5.6. Greedy and beam search both achieve reasonable BLEU scores but rely on a high number of search errors[15] to not be affected by a serious NMT model error: For 51.8% of the sentences, NMT assigns the global best model score to the empty translation, i.e. a single $< /\mathrm{s} >$ token. Fig. 5.7 visualizes the relationship between BLEU and the number of search errors. Large beam sizes reduce the number of search errors, but the BLEU score drops because translations are too short. Even a large beam size of 100 produces 53.62% search errors. Fig. 5.8 shows that beam search effectively reduces search errors with respect to greedy decoding to some degree, but is ineffective in reducing search errors even further. For example, Beam-10 yields 15.9% fewer search errors (absolute) than greedy decoding (57.68% vs. 73.58%), but Beam-100 improves search only slightly (53.62% search errors) despite being 10 times slower than beam-10.

[15] A sentence is classified as search error if the decoder does not find the global best model score.

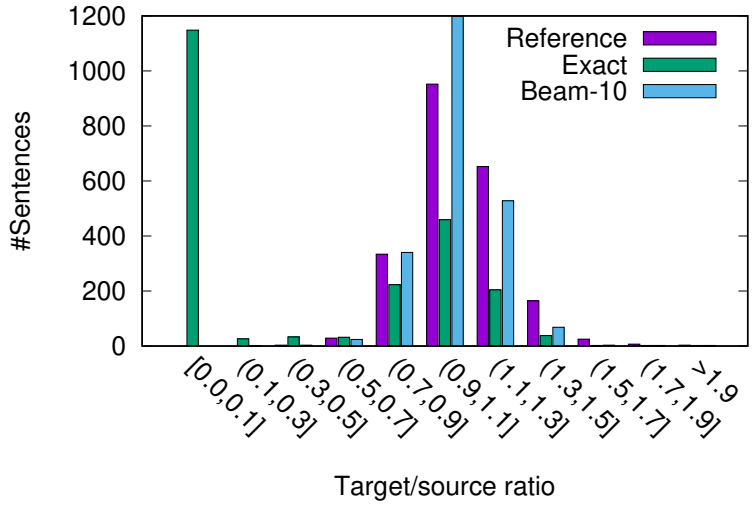


Fig. 5.9 Histogram over target/source length ratios.

Model	Beam-10		Exact
	BLEU	#Search err.	#Empty
LSTM*	28.6	58.4%	47.7%
SliceNet*	28.8	46.0%	41.2%
Transformer-Base	30.3	57.7%	51.8%
Transformer-Big*	31.7	32.1%	25.8%

Table 5.7 *: The recurrent LSTM, the convolutional SliceNet (Kaiser et al., 2017), and the Transformer-Big systems are strong baselines from our WMT'18 shared task submission (Sec. 4.6).

The problem of empty translations is also visible in the histogram over length ratios (Fig. 5.9). Beam search – although still slightly too short – roughly follows the reference distribution, but exact search has an isolated peak in $[0.0, 0.1]$ from the empty translations.

Tab. 5.7 demonstrates that the problems of search errors and empty translations are not specific to the Transformer base model and also occur with other architectures. Even a highly optimized Transformer Big model from our WMT18 shared task submission (Sec. 4.6) has 25.8% empty translations.

Fig. 5.10 shows that long source sentences are more affected by both beam search errors and the problem of empty translations. The global best translation is empty for almost all sentences longer than 40 tokens (green curve). Even without sentences where the model prefers the empty translation, a large amount of search errors remain (blue curve).

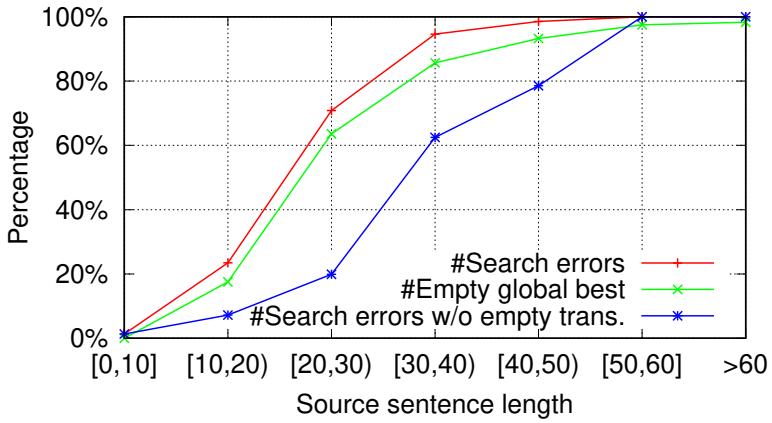


Fig. 5.10 Number of search errors under Beam-10 and empty global bests over the source sentence length.

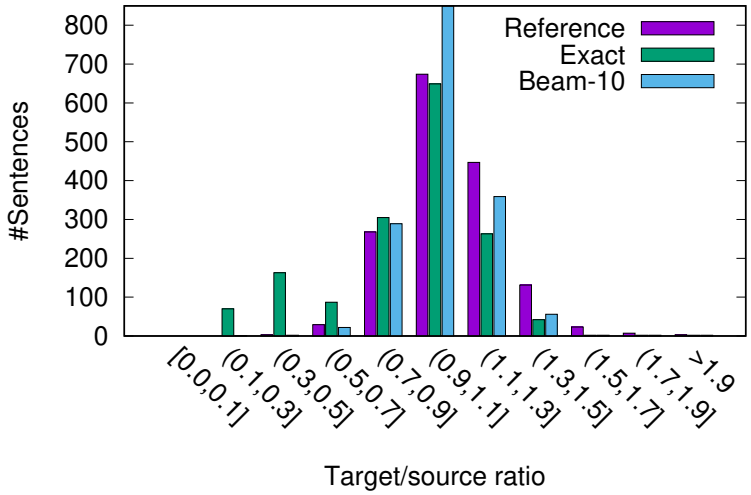


Fig. 5.11 Histogram over length ratios with minimum translation length constraint of 0.25 times the source sentence length. Experiment conducted on 73.0% of the test set.

Results with Length Constraints To find out more about the length deficiency we constrained exact search to certain translation lengths. Constraining search that way increases the run time as the γ-bounds are lower. Therefore, all results in this section are conducted on only a subset of the test set to keep the runtime under control.[16] We first constrained search to translations longer than 0.25 times the source sentence length and thus excluded the empty translation from the search space. Although this mitigates the problem slightly (Fig. 5.11), it still results in a peak in the $(0.3, 0.5]$ cluster. This suggests that the problem of

[16] We stopped decoding if the decoder took longer than a day for a single sentence on a single CPU. Exact search *without* length constraints is much faster and does not need maximum execution time limits.

Search	BLEU	Ratio
Beam-10	37.0	1.00
Exact for Beam-10 length	37.0	1.00
Exact for reference length	37.9	1.01

Table 5.8 Exact search under length constraints. Experiment conducted on 48.3% of the test set.

Search	W/o length norm.		With length norm.	
	BLEU	Ratio	BLEU	Ratio
Beam-10	37.0	1.00	36.3	1.03
Beam-30	36.7	0.98	36.3	1.04
Exact	27.2	0.74	36.4	1.03

Table 5.9 Length normalization fixes translation lengths, but prevents exact search from matching the BLEU score of Beam-10. Experiment conducted on 48.3% of the test set.

empty translations is the consequence of an inherent model bias towards shorter hypotheses and cannot be fixed with a length constraint.

We then constrained exact search to either the length of the best Beam-10 hypothesis or the reference length. Tab. 5.8 shows that exact search constrained to the Beam-10 hypothesis length does not improve over beam search, suggesting that any search errors between beam search score and global best score for that length are insignificant enough so as not to affect the BLEU score. The oracle experiment in which we constrained exact search to the correct reference length (last row in Tab. 5.8) improved the BLEU score by 0.9 points.

A popular method to counter the length bias in NMT is *length normalization* (Boulanger-Lewandowski et al., 2013; Jean et al., 2015b) which simply divides the sentence score by the sentence length. We can find the global best translations under length normalization by generalizing our exact inference scheme to *length dependent* lower bounds γ_k. The generalized scheme[17] finds the best model scores for each translation length k in a certain range (e.g. zero to 1.2 times the source sentence length). The initial lower bounds are derived from the Beam-10 hypothesis $\mathbf{y}_{\text{beam}}$ as follows:[18]

$$\gamma_k = (k+1)\frac{\log P(\mathbf{y}_{\text{beam}}|\mathbf{x})}{|\mathbf{y}_{\text{beam}}|+1}. \tag{5.5}$$

[17] Available in our SGNMT decoder as `simplelendfs` strategy.

[18] We add 1 to the lengths to avoid division by zero errors.

Exact search under length normalization does not suffer from the length deficiency anymore (last row in Tab. 5.9), but it is not able to match our best BLEU score under Beam-10 search. This suggests that while length normalization biases search towards translations of roughly the correct length, it does not fix the fundamental modelling problem.

Conclusion We have presented an exact inference scheme for NMT. Exact search may not be practical, but it allowed us to discover deficiencies in widely used NMT models. We linked deteriorating BLEU scores of large beams with the reduction of search errors and showed that the model often prefers the empty translation – an evidence of NMT's failure to properly model adequacy. Our investigations into length constrained exact search suggested that simple heuristics like length normalization are unlikely to remedy the problem satisfactorily.

5.5 Output Formats

SGNMT aims to find a complete hypothesis which is scored highly by the predictors. Many search procedures visit a number of complete hypotheses during that process. SGNMT supports six different output formats that write complete hypotheses to the file system:

- `text`: Plain text file with first best translations.
- `nbest`: n-best list of translation hypotheses.
- `sfst`: Lattice generation in OpenFST (Allauzen et al., 2007) format with standard arcs.
- `fst`: Lattices with sparse tuple arcs (Iglesias et al., 2015) which keep predictor scores separate.
- `ngram`: MBR-style n-gram posteriors as discussed in Secs. 4.5 and 4.6.
- `timecsv`: Detailed CSV file that contains predictor scores and weights at each time step.

5.6 Tuning Predictor Weights

SGNMT decodes according a linear combination of the predictor scores. It is often crucial to tune the predictor interpolation weights because predictor scores may be in a very different dynamic range. For example, NMT operates with log-likelihoods and thus always produces negative scores, but SMT scores are by themselves linear combinations of a large number of features and are normally not bounded. Furthermore, some predictors (e.g. NMT) may be more

Algorithm 12 Optimize($f : [0,1] \to \mathbb{R}_+$)

$\mathscr{E} = \{(0,0), (0.5, f(0.5)), (1,0)\}$
repeat
  $(\text{mid}, \_) \leftarrow \arg\max_{(p,q)\in\mathscr{E}} q$
  $\text{lo}' \leftarrow \max_{(p,q)\in\mathscr{E}:p<\text{mid}} p$
  $\text{hi}' \leftarrow \min_{(p,q)\in\mathscr{E}:p>\text{mid}} p$
  $\text{lo} \leftarrow \frac{\text{mid}+\text{lo}'}{2}$
  $\text{hi} \leftarrow \frac{\text{mid}+\text{hi}'}{2}$
  $\mathscr{E} \leftarrow \mathscr{E} \cup \{(\text{lo}, f(\text{lo})), (\text{hi}, f(\text{hi}))\}$
until Convergence
return $\arg\max_{(p,q)\in\mathscr{E}} q$

informative than others (e.g. LM) which should be reflected by the interpolation weights. All our methods for hybrid SMT-NMT in Ch. 4 require predictor weight tuning (usually denoted with λ in Ch. 4).

Feature weights in traditional SMT are usually tuned iteratively based on a minimum error rate training (MERT) criterion: The system decodes with the current set of weights and generates an n-best list (Och, 2003) or a lattice (Macherey et al., 2008). The weights are optimized by maximizing BLEU on this (rich) SMT output. In the next iteration, SMT generates new n-best lists or lattices with the updated weights.

Tuning is an important and well-studied part of the traditional SMT training pipeline. It is therefore tempting and, using SGNMT's `nbest` or `fst` output formats, technically possible to adapt methods developed for SMT to tuning predictor weights. However, in our experiments, n-best or lattice MERT for SGNMT did not work well when neural sequence models are involved. The most likely problem is diversity: SMT can generate large and diverse lattices, whereas

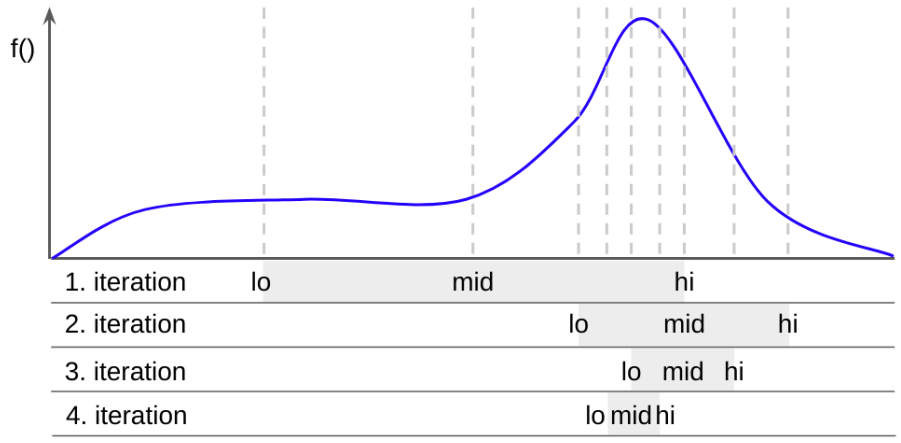


Fig. 5.12 Line search algorithm for SGNMT tuning.

n-best lists from neural beam search are shallow and relatively homogeneous (Sec. 3.7.7). All of our more recent research work therefore uses a tuning scheme based on the 1-best translation only. Our tuning algorithm is an instance of Powell's method (Powell, 1964) that optimizes one feature weight at a time while keeping all other weights fixed. Powell's method is more a template than a a complete algorithm: It reduces the multimodal optimization of interpolation weights to a series of unimodal optimizations along single dimensions. These isolated unimodal optimizations are handled by a *line search* algorithm. We propose a new line search algorithm that is inspired by golden-section search (Kiefer, 1953) and illustrated in Fig. 5.12. Alg. 12 gives a formal description in terms of maximizing a non-differentiable function $f(\cdot)$ (in our case usually BLEU) over the interval $[0, 1]$. The algorithm keeps a set of explored function points $\mathscr{E}$. In each iteration it explores two new points (lo and hi) left and right of the best value so far (mid). If used inside Powell's method we keep $\mathscr{E}$ between iterations, but add some slack if the feature weight vector has been updated along another dimension.

5.7 Applications

5.7.1 SGNMT for Research

SGNMT is designed for environments in which implementation time is far more valuable than computation time. This basic design decision is strongly reflected by the software architecture which accepts degradations in runtime in favor of extensibility and flexibility. We designed SGNMT that way because training models and coding usually take the most time in our day-to-day work. Decoding, however, usually takes a small fraction of that time. Therefore, reducing the implementation time has a much larger impact on the overall productivity of our research group than improvements in runtime, especially since decoding can be easily parallelized on multiple machines.

Another benefit of SGNMT's predictor framework is that it enables us to write code independently of any NMT package, and swap the NMT back end with more recent software if needed. For example, our previous research work on lattice rescoring (Sec. 4.3) and MBR-based NMT (Sec. 4.5) used the NMT package Blocks (van Merriënboer et al., 2015) which is based on Theano (Bastien et al., 2012). Since both Blocks and Theano have been discontinued, we recently switched to a Tensor2Tensor (Vaswani et al., 2018) back-end based on TensorFlow (Abadi et al., 2016). Without reimplementation, we could validate that MBR-based NMT holds up even under a much stronger NMT model, the Transformer model (Vaswani et al., 2017). Tab. 5.10 compares the performance of lattice rescoring and MBR-based combination across four different NMT implementations using SGNMT.

	Pure NMT	SMT lattice rescoring	MBR-based NMT-SMT hybrid
Theano: Blocks (van Merriënboer et al., 2015)	18.4	18.9	19.0
TensorFlow: seq2seq tutorial[19]	17.5	19.3	19.2
TensorFlow: NMT tutorial[20]	18.8	19.1	20.0
TensorFlow: T2T Transformer (Vaswani et al., 2018)	21.7	19.3	22.5

Table 5.10 BLEU scores of SGNMT with different NMT back ends on the complete KFTT test set (Neubig, 2011) computed with `multi-bleu.pl`. All neural systems are BPE-based (Sennrich et al., 2016c) with vocabulary sizes of 30K. The SMT baseline achieves 18.1 BLEU.

5.7.2 SGNMT for Teaching

SGNMT is being used for teaching at the University of Cambridge in course work and student research projects. Several students on the Cambridge MPhil in Machine Learning and Machine Intelligence and Master students used SGNMT for their dissertation projects. Gao (2016)'s project involved using SGNMT with OpenFST (Allauzen et al., 2007) for applying subword models in SMT (Gao, 2016). Tomczak (2016) developed automatic music composition by LSTMs where WFSAs were used to define the space of allowable chord progressions in 'Bach' chorales. The LSTM provides the 'creativity' and the WFSA enforces constraints on chord progressions that the chorales must obey. Wang (2018) used SGNMT for simultaneous neural machine translation. Kell (2018) tried to use the Bayesian interpolation implementation in SGNMT to overcome catastrophic forgetting – a line of research which motivated Saunders et al. (2019) to study catastrophic forgetting in NMT in a broader context using SGNMT. This year, two further student theses will make use of SGNMT, one on grammatical error correction and one on knowledge distillation.

SGNMT is also part of three practicals for MPhil students at Cambridge. The first practical applies different kinds of language models to restore the correct casing in a lowercased sentence using FSTs. Since SGNMT has good support for the OpenFST library (Allauzen et al., 2007) and can both read and write FSTs, it is used to integrate neural models such as RNN LMs into the exercise. The second practical focuses on decoding strategies for NMT and explores the synergies of word- and subword-based models and the potential of combining SMT and NMT. The third practical teaches students how neural models and FSTs can be combined to tackle the problem of grammatical error correction, the automatic correction of errors in text. This approach to grammatical error correction will be discussed in detail in Sec. 7.3.

[19] https://github.com/ehasler/tensorflow
[20] https://github.com/tensorflow/nmt, trained with Tensor2Tensor (Vaswani et al., 2018)

5.7.3 SGNMT in the Industry

SGNMT has been adapted by SDL Research for rapid prototyping and assessment of new research avenues, including attention-based NMT, model shrinking (Ch. 6), MBR-based NMT (Sec. 4.5), and the Transformer (Vaswani et al., 2017) architecture. For more details about how SDL Research uses SGNMT in its processes we refer to Iglesias et al. (2018); Stahlberg et al. (2018d).

5.8 Conclusion

This chapter focused on the SGNMT decoding platform – the software backbone for most of the experiments in this thesis. SGNMT's software architecture consisting of *predictors* and *decoders* is designed to minimize the implementation time required for adding new search algorithms or scoring strategies, and is thus particularly suitable for quick prototyping of new research ideas and painless transitioning to new machine learning frameworks and NMT tools. SGNMT has been used for most of the current research work of the Cambridge MT group, and has been adapted for teaching and prototyping in the industry. The exact inference schemes in SGNMT have led us to a reassessment of the amount of NMT search and model errors (Sec. 5.4.4). Paradoxically, NMT often assigns the global best score to the empty translation, and relies on beam search errors in practice to not find this global best translation.

I may not have gone where I intended to go, but I think I have ended up where I needed to be.

Douglas Adams

6 Unfolding and Shrinking Neural Machine Translation Ensembles

This chapter was previously published as paper (Stahlberg and Byrne, 2017), but has been extended and modified since then for a seamless integration into this thesis.

6.1 Motivation

The top systems in recent machine translation evaluation campaigns on various language pairs use ensembles of a number of NMT systems (see Sec. 3.7.4). Ensembling (Dietterich, 2000; Hansen and Salamon, 1990) of neural networks is a simple yet very effective technique to improve the accuracy of NMT. The decoder makes use of K NMT networks which are either trained independently (Chung et al., 2016; Neubig, 2016; Sutskever et al., 2014; Wu et al., 2016b) or share some amount of training iterations (Cromieres et al., 2016; Durrani et al., 2016; Sennrich et al., 2016a,c). The ensemble decoder computes predictions from each of the individual models which are then combined using the arithmetic average (Sutskever et al., 2014) or the geometric average (Cromieres et al., 2016).

Ensembling consistently outperforms single NMT by a large margin. However, the decoding speed is significantly worse since the decoder needs to apply K NMT models rather than only

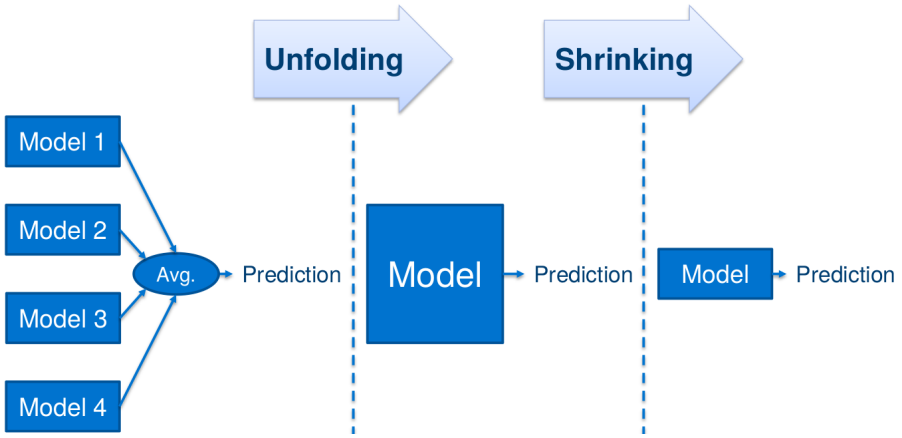
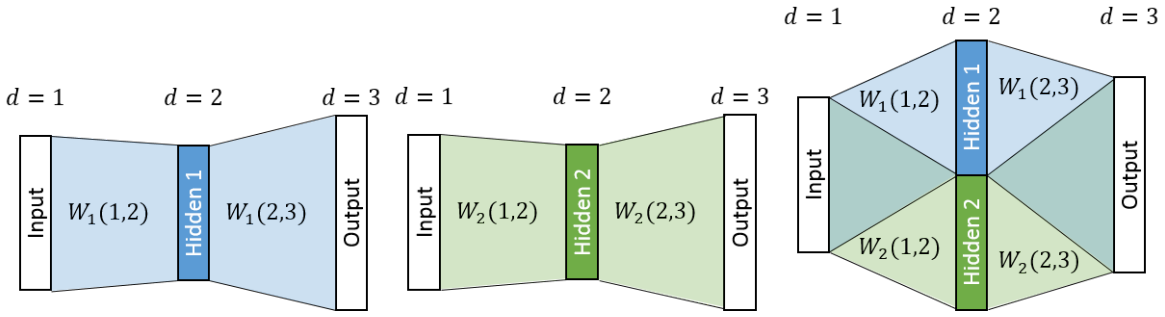


Fig. 6.1 Unfolding and shrinking as a combined approach to efficient ensembling.

one. Therefore, a recent line of research transfers the idea of *knowledge distillation* (Sec. 3.16) to NMT and trains a smaller network (the student) by minimizing the cross-entropy to the output of the ensemble system (the teacher). This chapter presents an alternative to knowledge distillation as we aim to speed up decoding to be comparable to single NMT while retaining the boost in translation accuracy from the ensemble. In a first step, we describe how to construct a single large neural network which imitates the output of an ensemble of multiple networks with the same topology. We will refer to this process as *unfolding*. GPU-based decoding with the unfolded network is often much faster than ensemble decoding since more work can be done on the GPU. In a second step, we explore methods to reduce the size of the unfolded network. Fig. 6.1 illustrates our approach. This idea is justified by the fact that ensembled neural networks are often over-parameterized and have a large degree of redundancy (Hassibi et al., 1993; LeCun et al., 1989b; Srinivas and Babu, 2015). Shrinking the unfolded network leads to a smaller model which consumes less space on the disk and in the memory; a crucial factor on mobile devices. More importantly, the decoding speed on all platforms benefits greatly from the reduced number of neurons. We find that the dimensionality of linear embedding layers in the NMT network can be reduced heavily by low-rank matrix approximation based on singular value decomposition (SVD). This suggest that high dimensional embedding layers may be needed for training, but do not play an important role for decoding. The NMT network, however, also consists of complex layers like gated recurrent units (Cho et al., 2014b, GRUs) and attention (Bahdanau et al., 2015). Therefore, we introduce a novel algorithm based on linear combinations of neurons which can be applied either during training (*data-bound*) or directly on the weight matrices without using training data (*data-free*). We report that with a mix of the presented shrinking methods we are able to reduce the size of the unfolded network to the size of the single NMT network while keeping the boost in BLEU score from the ensemble.



(a) Single network 1. (b) Single network 2. (c) Unfolded network.

Fig. 6.2 Unfolding mimics the output of the ensemble of two single layer feedforward networks.

Depending on the aggressiveness of shrinking, we report either a gain of 2.2 BLEU at the same decoding speed, or a 3.4× CPU decoding speed up with only a minor drop in BLEU compared to the original single NMT system. Furthermore, it is often much easier to stage a single NMT system than an ensemble in a commercial MT workflow, and it is crucial to be able to optimize quality at specific speed and memory constraints. Unfolding and shrinking address these problems directly.

In this chapter we exclusively focus on recurrent NMT models such as RNNsearch (Sec. 3.6.3) as decoding speed tends to be slow in recurrent models due to sequential computation.

6.2 Unfolding Ensembles into a Single Large Neural Network

The first concept of our approach is called *unfolding*. Unfolding is an alternative to ensembling of multiple neural networks with the same topology. Rather than averaging their predictions, unfolding constructs a single large neural net out of the individual models which has the same number of input and output neurons but larger inner layers. Our main motivation for unfolding is to obtain a single network with ensemble level performance which can be shrunk with the techniques in Sec. 6.3.

Suppose we ensemble two single layer feedforward neural nets as shown in Fig. 6.2. Normally, ensembling is implemented by performing an isolated forward pass through the first network (Fig. 6.2a), another isolated forward pass through the second network (Fig. 6.2b), and averaging the activities in the output layers of both networks. This can be simulated by merging both networks into a single large network as shown in Fig. 6.2c. The first neurons in the hidden layer of the combined network correspond to the hidden layer in the first single network, and

the others to the hidden layer of the second network. A single pass through the combined network yields the same output as the ensemble if the output layer is linear (up to a factor 2). The weight matrices in the unfolded network can be constructed by stacking the corresponding weight matrices (either horizontally or vertically) in network 1 and 2. This kind of aggregation of multiple networks with the same topology is not only possible for single-layer feedforward architectures but also for complex networks consisting of multiple GRU layers and attention.

For a formal description of unfolding we address layers with indices $d = 0, 1, \ldots, D$. The special layer 0 has a single neuron for modelling bias vectors. Layer 1 holds the input neurons and layer D is the output layer. We denote the size of a layer in the individual models as $s(d)$. When combining K networks, the layer size $s'(d)$ in the unfolded network is increased by factor K if d is an inner layer, and equal to $s(d)$ if d is the input or output layer.

$$s'(d) = \begin{cases} Ks(d) & \text{if } d \in [2, D-1] \\ s(d) & \text{otherwise} \end{cases} \tag{6.1}$$

We denote the weight matrix between two layers $d_1, d_2 \in [0, D]$ in the k-th individual model ($k \in [1, K]$) as $W_k(d_1, d_2) \in \mathbb{R}^{s(d_1) \times s(d_2)}$, and the corresponding weight matrix in the unfolded network as $W'(d_1, d_2) \in \mathbb{R}^{s'(d_1) \times s'(d_2)}$. We explicitly allow d_1 and d_2 to be non-consecutive or reversed to be able to model recurrent networks. We use the zero-matrix if layers d_1 and d_2 are not connected. The construction of the unfolded weight matrix $W'(d_1, d_2)$ from the individual matrices $W_k(d_1, d_2)$ depends on whether the connected layers are inner layers or not. A complete formula which describes how to build $W'(d_1, d_2)$ follows.

$$W'(d_1, d_2) = \begin{cases} \begin{pmatrix} W_1(d_1,d_2) & 0 & \cdots & 0 \\ 0 & W_2(d_1,d_2) & \vdots & \vdots \\ \vdots & \cdots & \ddots & 0 \\ 0 & \cdots & & W_K(d_1,d_2) \end{pmatrix} & \text{if } d_1 \in \text{Hidden}, d_2 \in \text{Hidden} \\ \frac{1}{K} \begin{pmatrix} W_1(d_1,d_2) \\ \vdots \\ W_K(d_1,d_2) \end{pmatrix} & \text{if } d_1 \in \text{Hidden}, d_2 \notin \text{Hidden} \\ \begin{pmatrix} W_1(d_1,d_2) & \cdots & W_K(d_1,d_2) \end{pmatrix} & \text{if } d_1 \notin \text{Hidden}, d_2 \in \text{Hidden} \end{cases} \tag{6.2}$$

where Hidden $:= [2, D-1]$ denotes the set of all inner layers. Unfolded NMT networks approximate but do not exactly match the output of the ensemble due to two reasons. First, the unfolded network synchronizes the attentions of the individual models. Each decoding

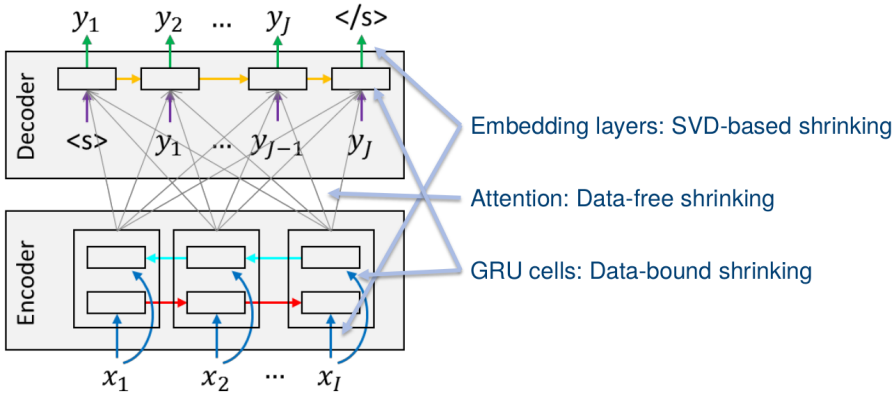


Fig. 6.3 Data-free, data-bound, and SVD-based shrinking.

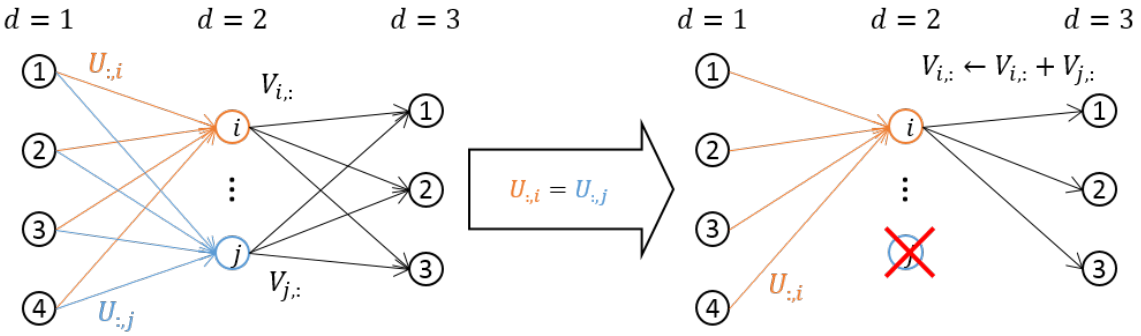
step in the unfolded network computes a single attention weight vector. In contrast, ensemble decoding would compute one attention weight vector for each of the K input models. A second difference is that the ensemble decoder first applies the softmax at the output layer, and then averages the prediction probabilities. The unfolded network averages the neuron activities (i.e. the logits) first, and then applies the softmax function. Interestingly, as shown in Sec. 6.4, these differences did not have any impact on the BLEU score but yield potential speed advantages of unfolding since the computationally expensive softmax layer is only applied once.

6.3 Shrinking the Unfolded Network

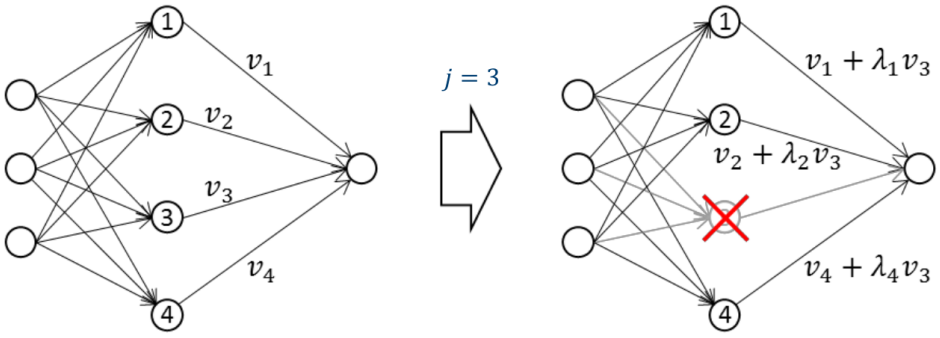
After constructing the weight matrices of the unfolded network we reduce the size of it by iteratively shrinking layer sizes. In this section we denote the incoming weight matrix of the layer to shrink as $U \in \mathbb{R}^{m_{in} \times m}$ and the outgoing weight matrix as $V \in \mathbb{R}^{m \times m_{out}}$. As shown in Fig. 6.3, we use three different methods to shrink different parts of the NMT network: data-free, data-bound, and SVD-based shrinking.

6.3.1 Data-Free Neuron Removal

Our data-free shrinking procedure is inspired by the method of Srinivas and Babu (2015). They propose a criterion for removing neurons in inner layers of the network based on two intuitions. First, similarly to Hebb's learning rule (Hebb, 1949), they detect redundancy by the principle *neurons which fire together, wire together*. The idea is illustrated in Fig. 6.4a. If the incoming weight vectors $U_{:,i}$ and $U_{:,j}$ are exactly the same for two neurons i and j, we can remove the neuron j and add its outgoing connections to neuron i ($V_{i,:} \leftarrow V_{i,:} + V_{j,:}$) without changing the



(a) Intuition behind the method of Srinivas and Babu (2015) (adapted from Fig. 1 in (Srinivas and Babu, 2015)).



(b) Intuition behind our data-free and data-bound methods.

Fig. 6.4 Neuron removal methods.

output.[1] This holds since the activity in neuron j will always be equal to the activity in neuron i. In practice, Srinivas and Babu (2015) use a distance measure based on the difference of the incoming weight vectors to search for similar neurons as exact matches are very rare.

The second intuition of the criterion used by Srinivas and Babu (2015) is that neurons with small outgoing weights contribute very little overall. Therefore, they search for a pair of neurons $i, j \in [1, m]$ according the following term and remove the j-th neuron.[2]

$$\underset{i,j \in [1,m]}{\arg\min} ||U_{:,i} - U_{:,j}||_2^2 ||V_{j,:}||_2^2 \tag{6.3}$$

Neuron j is selected for removal if (1) there is another neuron i which has a very similar set of incoming weights and if (2) j has a small outgoing weight vector. Their criterion is *data-free* since it does not require any training data. For further details we refer to Srinivas and Babu (2015).

[1] We denote the i-th row vector of a matrix A with $A_{i,:}$ and the i-th column vector as $A_{:,i}$ as defined in Sec. 1.4.

[2] Note that the criterion in Eq. 6.3 generalizes the criterion of Srinivas and Babu (2015) to multiple outgoing weights. Also note that Srinivas and Babu (2015) propose some heuristic improvements to this criterion. However, these heuristics did not work well in our NMT experiments.

Srinivas and Babu (2015) propose to add the outgoing weights of j to the weights of a similar neuron i to compensate for the removal of j. However, we have found that this approach does not work well on NMT networks. We propose instead to compensate for the removal of a neuron by a linear combination of the remaining neurons in the layer (Fig. 6.4b). Data-free shrinking assumes for the sake of deriving the update rule that the neuron activation function is linear. We now ask the following question: How can we compensate as well as possible for the loss of neuron j such that the impact on the output of the whole network is minimized? Data-free shrinking represents the incoming weight vector of neuron j ($U_{:,j}$) as linear combination of the incoming weight vectors of the other neurons. The linear factors can be found by satisfying the following linear system:

$$U_{:,\neg j}\lambda = U_{:,j} \tag{6.4}$$

where $U_{:,\neg j}$ is matrix U without the j-th column. In practice, we use the method of ordinary least squares to find the λ-vector because the system may be overdetermined. The idea is that if we mix the outputs of all neurons in the layer by the λ-weights, we get the output of the j-th neuron. The row vector $V_{j,:}$ contains the contributions of the j-th neuron to each of the neurons in the next layer. Rather than using these connections, we approximate their effect by adding some weight to the outgoing connections of the other neurons. How much weight depends on λ and the outgoing weights $V_{j,:}$. The factor $D_{k,l}$ which we need to add to the outgoing connection of the k-th neuron to compensate for the loss of the j-th neuron on the l-th neuron in the next layer is:

$$D_{k,l} = \lambda_k V_{j,l} \tag{6.5}$$

Therefore, the update rule for V is:

$$V \leftarrow V + D \tag{6.6}$$

In the remainder we will refer to this method as *data-free* shrinking. Note that we recover the update rule of Srinivas and Babu (2015) by setting λ to the i-th unit vector. Also note that the error introduced by our shrinking method is due to the fact that we ignore the non-linearity, and that the solution for λ may not be exact. The method is error-free on linear layers as long as the residuals of the least-squares analysis in Eq. 6.4 are zero.

GRU layers The terminology of *neurons* needs some further elaboration for GRU layers which rather consist of update and reset gates and states (Cho et al., 2014b). On GRU layers, we treat the states as neurons, i.e. the j-th neuron refers to the j-th entry in the GRU state vector. Input connections to the gates are included in the incoming weight matrix U for estimating λ in Eq. 6.4. Removing neuron j in a GRU layer means deleting the j-th entry in the states and both gate vectors.

6.3.2 Data-Bound Neuron Removal

Although we find our data-free approach to be a substantial improvement over the methods of Srinivas and Babu (2015) on NMT networks, it still leads to a non-negligible decline in BLEU score when applied to recurrent GRU layers. Our data-free method uses the incoming weights to identify similar neurons, i.e. neurons expected to have similar activities. This works well enough for simple layers, but the interdependencies between the states and the gates inside gated layers like GRUs or LSTMs are complex enough that redundancies cannot be found simply by looking for similar weights. In the spirit of Babaeizadeh et al. (2016), our *data-bound* version records neuron activities during training to estimate λ. We compensate for the removal of the j-th neuron by using a linear combination of the output of remaining neurons with similar activity patterns. In each layer, we prune 40 neurons each 450 training iterations until the target layer size is reached. Let A be the matrix which holds the records of neuron activities in the layer since the last removal. For example, for the decoder GRU layer, a batch size of 80, and target sentence lengths of 20, A has $20 \cdot 80 \cdot 450 = 720K$ rows and m (the number of neurons in the layer) columns.[3] Similarly to Eq. 6.4 we find interpolation weights λ using the method of least squares on the following linear system.

$$A_{:,\neg j}\lambda = A_{:,j} \tag{6.7}$$

The update rule for the outgoing weight matrix is the same as for our data-free method (Eq. 6.6). The key difference between data-free and data-bound shrinking is the way λ is estimated. Data-free shrinking uses the similarities between incoming weights, and data-bound shrinking uses neuron activities recorded during training. Once we select a neuron to remove, we estimate λ, compensate for the removal, and proceed with the shrunk network. Both methods are prior to any decoding and result in shrunk parameter files which are then loaded to the decoder. Both methods remove neurons rather than single weights.

The data-bound algorithm runs gradient-based optimization on the unfolded network. We use the AdaGrad (Duchi et al., 2011) step rule, a small learning rate of 0.0001, and aggressive step clipping at 0.05 to avoid destroying useful weights which were learned in the individual networks prior to the construction of the unfolded network.

Our data-bound algorithm uses a data-bound version of the neuron selection criterion in Eq. 6.3 which operates on the activity matrix A. We search for the pair $i, j \in [1, m]$ according the following term and remove neuron j.

[3] In practice, we use a random sample of 50K rows rather than the full matrix to keep the complexity of the least-squares analysis under control.

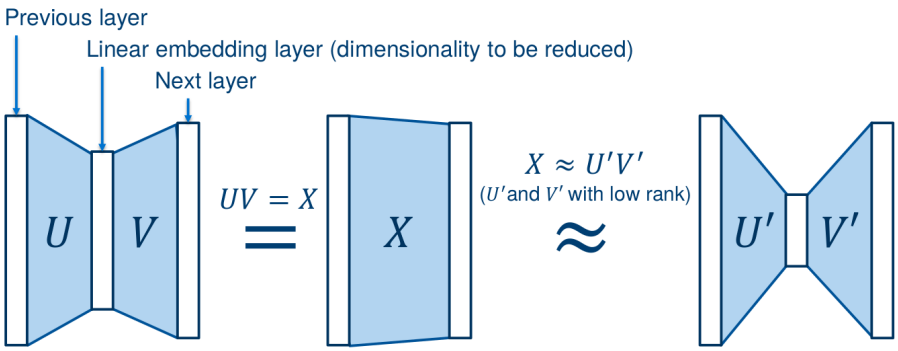


Fig. 6.5 SVD-based shrinking.

$$\arg\min_{i,j\in[1,m]} ||A_{:,i} - A_{:,j}||_2^2 ||A_{:,j}||_2^2 \tag{6.8}$$

6.3.3 Shrinking Embedding Layers with SVD

The standard attention-based recurrent NMT network architecture (Bahdanau et al., 2015) includes three linear layers: the embedding layer in the encoder, and the output and feedback embedding layers in the decoder. We have found that linear layers are particularly easy to shrink using low-rank matrix approximation. As before we denote the incoming weight matrix as $U \in \mathbb{R}^{m_{in} \times m}$ and the outgoing weight matrix as $V \in \mathbb{R}^{m \times m_{out}}$. Since the layer is linear, we could directly connect the previous layer with the next layer using the product of both weight matrices $X = U \cdot V$. However, X may be very large. Therefore, we approximate X as a product of two low rank matrices $U' \in \mathbb{R}^{m_{in} \times m'}$ and $V' \in \mathbb{R}^{m' \times m_{out}}$ ($X \approx U'V'$) where $m' \ll m$ is the desired layer size. A very common way to find such a matrix factorization is using truncated singular value decomposition (SVD). The layer is eventually shrunk by replacing U with U' and V with V'. SVD-based shrinking is illustrated in Fig. 6.5.

6.4 Results

The individual NMT systems we use as source for constructing the unfolded networks are trained using AdaDelta (Zeiler, 2012) on the Blocks/Theano implementation (Bastien et al., 2012; van Merriënboer et al., 2015) of the standard attention-based recurrent NMT model (Bahdanau et al., 2015) with: 1000 dimensional GRU layers (Cho et al., 2014b) in both the decoder and bidirectional encoder; a single maxout output layer (Goodfellow et al., 2013b); and 620 dimensional embedding layers. We follow Sennrich et al. (2016c) and use subword units based

	Base	Shrinking Methods: Embed.	GRUs	Attention	Maxout	Size Factor	BLEU: dev	test
(a)	Single	-	-	-	-	1.00	20.8	23.5
(b)	2-Ens.	-	-	-	-	2×1.00	22.7	25.2
(c)	2-Unfold	SVD	-	-	-	1.85	22.7	25.1
(d)	2-Unfold	SVD	-	-	-	1.77	22.7	25.1
(e)	2-Unfold	SVD	Data-Free	Data-Free	-	1.05	21.6	24.2
(f)	2-Unfold	SVD	Data-Bound	Data-Free	-	1.05	22.4	**25.3**
(g)	2-Unfold	SVD	Data-Bound	Data-Free	Data-Free	1.00	16.9	19.3
(h)	2-Unfold	SVD	Data-Bound	Data-Free	Data-Bound	1.00	21.9	24.6

Table 6.1 Shrinking layers of the unfolded network on Ja-En to their original size.

on byte pair encoding rather than words as modelling units. Our SGNMT decoder[4] (Ch. 5) with a beam size of 12 is used in all experiments. Our primary corpus is the Japanese-English (Ja-En) ASPEC data set (Nakazawa et al., 2016). We select a subset of 500K sentence pairs to train our models as suggested by Neubig et al. (2015). We report cased BLEU scores calculated with Moses' `multi-bleu.pl` to be strictly comparable to the evaluation done in the Workshop of Asian Translation (WAT). We also apply our method to the WMT data set for English-German (En-De), using the *news-test2014* as a development set, and keeping *news-test2015* and *news-test2016* as test sets. En-De BLEU scores are computed using `mteval-v13a.pl` as in the WMT evaluation. We set the vocabulary sizes to 30K for Ja-En and 50K for En-De. We also report the *size factor* for each model which is the total number of model parameters (sum of all weight matrix sizes) divided by the number of parameters in the original NMT network (86M for Ja-En and 120M for En-De). We choose a widely used, simple ensembling method (prediction averaging) as our baseline. We feel that the prevalence of this method makes it a reasonable baseline for our experiments.

Shrinking the Unfolded Network First, we investigate which shrinking methods are effective for which layers. Tab. 6.1 summarizes our results on a 2-unfold network for Ja-En, i.e. two separate NMT networks are combined in a single large network as described in Sec. 6.2. The layers in the combined network are shrunk to the size of the original networks using the methods discussed in Sec. 6.3.

Shrinking the linear embedding layers with SVD (Sec. 6.3.3) is very effective. The unfolded model with shrunk embedding layers performs at the same level as the ensemble (compare

[4] 'vanilla' decoding strategy

	Compensation Method		BLEU	
	Linear Combination	SGD	dev	test
(a)			16.3	18.0
(b)	✓		22.1	24.3
(c)		✓	21.7	24.4
(d)	✓	✓	22.4	25.3

Table 6.2 Compensating for neuron removal in the data-bound algorithm. Row (d) corresponds to row (f) in Tab. 6.1.

rows (b) and (c)). In our initial experiments, we applied the method of Srinivas and Babu (2015) to shrink the other layers, but their approach performed very poorly on this kind of network: the BLEU score dropped down to 15.5 on the development set when shrinking all layers except the decoder maxout and embedding layers, and to 9.9 BLEU when applying their method only to embedding layers.[5] Row (e) in Tab. 6.1 shows that our data-free algorithm from Sec. 6.3.1 is better suited for shrinking the GRU and attention layers, leading to a drop of only 1 BLEU point compared to the ensemble (b) (i.e. 0.8 BLEU better than the single system (a)). However, using the data-bound version of our shrinking algorithm (Sec. 6.3.2) for the GRU layers performs best.[6] The shrunk model yields about the same BLEU score as the ensemble on the test set (25.2 in (b) and 25.3 in (f)). Shrinking the maxout layer remains more of a challenge (rows (g) and (h)), but the number of parameters in this layer is small. Therefore, shrinking all layers except the maxout layer leads to almost the same number of parameters (factor 1.05 in row (f)) as the original NMT network (a), and thus to a similar storage size, memory consumption, and decoding speed, but with a 1.8 BLEU gain. Based on these results we fix the shrinking method used for each layer for all remaining experiments as follows: We shrink linear embedding layers with our SVD-based method, GRU layers with our data-bound method, the attention layer with our data-free method, and do not shrink the maxout layer.

Our data-bound algorithm from Sec. 6.3.2 has two mechanisms to compensate for the removal of a neuron. First, we use a linear combination of the remaining neurons to update the outgoing weight matrix by imitating its activations (Eq. 6.6). Second, stochastic gradient descent (SGD) fine-tunes all weights during this process. Tab. 6.2 demonstrates that both mechanisms are crucial for minimizing the effect of shrinking on the BLEU score.

Decoding Speed Our testing environment is an Ubuntu 16.04 with Linux 4.4.0 kernel, 32 GB RAM, an Intel® Core i7-6700 CPU at 3.40 GHz and an Nvidia GeForce GTX Titan X

[5] Results with the original method of Srinivas and Babu (2015) are not included in Tab. 6.1.

[6] If we apply different methods to different layers of the same network, we first apply SVD-based shrinking, then the data-free method, and finally the data-bound method.

	System	Words/Min. CPU	Words/Min. GPU	Size Factor	BLEU dev	BLEU test
(a)	Single	323.4	2993.6	1.00	20.8	23.5
(b)	2-Ensemble	163.7	1641.1	2 × 1.00	22.7	25.2
(c)	2-Unfold, shrunk embed.& attention	157.2	2592.2	1.77	22.7	25.1
(d)	2-Unfold, shrunk all except maxout	308.3	2961.4	1.05	22.4	25.3
(e)	3-Ensemble	110.9	1158.2	3 × 1.00	23.4	25.9
(f)	3-Unfold, shrunk embed.& attention	95.4	2182.1	2.99	23.2	25.9
(g)	3-Unfold, shrunk all except maxout	301.6	3024.4	1.09	22.2	25.3

Table 6.3 Time measurements on Ja-En. Layers are shrunk to their size in the original NMT model.

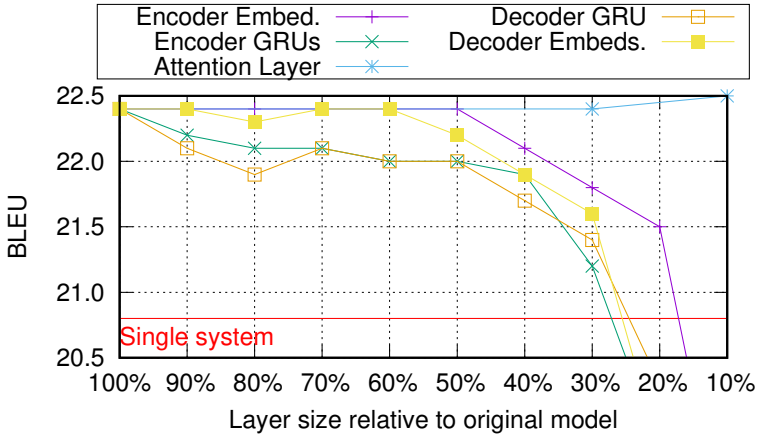


Fig. 6.6 Impact of shrinking on the BLEU score.

GPU. CPU decoding uses a single thread. We used the first 500 sentences of the Ja-En WAT development set for the time measurements.

Our results in Tab. 6.3 show that decoding with ensembles (rows (b) and (e)) is slow: combining the predictions of the individual models on the CPU is computationally expensive, and ensemble decoding requires K passes through the softmax layer which is also computationally expensive. Unfolding the ensemble into a single network and shrinking the embedding and attention layers improves the runtimes on the GPU significantly without noticeable impact on BLEU (rows (c) and (f)). This can be attributed to the fact that unfolding can reduce the communication overhead between CPU and GPU. Comparing rows (d) and (g) with row (a) reveals that shrinking the unfolded networks even further speeds up CPU and GPU decoding almost to the level of single system decoding. However, more aggressive shrinking yields a BLEU score of 25.3 when combining three systems (row (g)) – 1.8 BLEU better than the single

	Single	3-Unfold		
		Normal	Small	Tiny
Enc. Embed.	620	410	310	170
Enc. GRUs	1000	1300	580	580
Attention	1000	100	100	100
Dec. GRU	1000	1350	590	590
Dec. Maxout	500	1500	1500	1500
Dec. Embeds.	620	430	320	170
Size Factor	**1.00**	**1.00**	**0.50**	**0.33**

Table 6.4 Layer sizes of our setups for Ja-En.

system, but 0.6 BLEU worse than the 3-ensemble. Therefore, we will investigate the impact of shrinking on the different layers in the next sections more thoroughly.

Degrees of Redundancy in Different Layers We applied our shrinking methods to isolated layers in the 2-Unfold network of Tab. 6.1 (f). Fig. 6.6 plots the BLEU score when isolated layers are shrunk even below their size in the original NMT network. The attention layer is very robust against shrinking and can be reduced to 100 neurons (10% of the original size) without impacting the BLEU score. The embedding layers can be reduced to 60% but are sensitive to more aggressive pruning. Shrinking the GRU layers affects the BLEU score the most but still outperforms the single system when the GRU layers are shrunk to 30%.

Adjusting the Target Sizes of Layers Based on our previous experiments we revise our approach to shrink the 3-Unfold system in Tab. 6.3. Instead of shrinking all layers except the maxout layer to the same degree, we adjust the aggressiveness of shrinking for each layer. We suggest three different setups (*Normal*, *Small*, and *Tiny*) with the layer sizes specified in Tab. 6.4. *3-Unfold-Normal* has the same number of parameters as the original NMT networks (size factor: 1.0), *3-Unfold-Small* is only half their size (size factor: 0.5), and *3-Unfold-Tiny* reduces the size by two thirds (size factor: 0.33). When comparing rows (a) and (c) in Tab. 6.5 we observe that *3-Unfold-Normal* yields a gain of 2.2 BLEU with respect to the original single system and a slight improvement in decoding speed at the same time.[7] Networks with the size factor 1.0 like *3-Unfold-Normal* are very likely to yield about the same decoding speed as the *Single* network regardless of the decoder implementation, machine learning framework, and hardware. Therefore, we think that similar results are possible on other platforms as well.

[7]To validate that the gains come from ensembling and unfolding and not from the layer sizes in *3-Unfold-Normal* we trained a network from scratch with the same dimensions. This network performed similarly to our *Single* system.

	System	Words/Min. CPU	Words/Min. GPU	BLEU dev	BLEU test
(a)	Single	323.4	2993.6	20.8	23.5
(b)	3-Ensemble	110.9	1158.2	23.4	25.9
(c)	3-Unfold-Normal	445.2	3071.1	22.9	25.7
(d)	3-Unfold-Small	946.1	3572.0	21.7	23.9
(e)	3-Unfold-Tiny	1102.5	3483.7	20.6	23.2

Table 6.5 Our best models on Ja-En.

System	Wrds/Min. (GPU)	BLEU on news-test* 2014	2015	2016
Single	2128.7	19.6	21.9	24.6
2-Ensemble	1135.3	20.5	22.9	26.1
2-Unfold-Norm.	2099.1	20.7	23.1	25.8

Table 6.6 Our best models on En-De.

CPU decoding speed directly benefits even more from smaller setups – *3-Unfold-Tiny* is only 0.3 BLEU worse than *Single* but decoding on a single CPU is 3.4 times faster (row (a) vs. row (e) in Tab. 6.5). This is of great practical use: batch decoding with only two CPU threads surpasses production speed which is often set to 2000 words per minute (Beck et al., 2016). Our initial experiments in Tab. 6.6 suggest that the *Normal* setup is applicable to En-De as well, with substantial improvements in BLEU compared to *Single* with about the same decoding speed.

6.5 Probabilistic Interpretation of Data-Free and Data-Bound Shrinking

Data-free and data-bound shrinking can be interpreted as setting the expected difference between network outputs before and after a removal operation to zero under different assumptions.

For simplicity, we focus our probabilistic treatment of shrinking on single layer feedforward networks. Such a network maps an input $\mathbf{x} \in \mathbb{R}^{m_{in}}$ to an output $\mathbf{y} \in \mathbb{R}^{m_{out}}$. The l-th output y_l is computed according the following equation

$$y_l = \sum_{k \in [1,m]} \sigma(\mathbf{x}\mathbf{u}_k^T) V_{k,l} \tag{6.9}$$

where $\mathbf{u}_k \in \mathbb{R}^{m_{in}}$ is the incoming weight vector of the k-th hidden neuron (denoted as $U_{:,k}$ in previous sections) and $V \in \mathbb{R}^{m \times m_{out}}$ the outgoing weight matrix of the m-dimensional hidden layer. We now remove the j-th neuron in the hidden layer and modify the outgoing weights to compensate for the removal:

$$y'_l = \sum_{k \in [1,m] \setminus \{j\}} \sigma(\mathbf{x}\mathbf{u}_k^T) V'_{k,l} \tag{6.10}$$

where y'_l is the output after the removal operation and $V' \in \mathbb{R}^{m \times m_{out}}$ are the modified outgoing weights. Our goal is to choose V' such that the expected error introduced by removing neuron j is zero:

$$\mathbb{E}_{\mathbf{x}}(y_l - y'_l) = 0 \tag{6.11}$$

Data-free Shrinking Data-free shrinking makes two assumptions to satisfy Eq. 6.11. First, we assume that the incoming weight vector $\mathbf{u}_j$ can be represented as a linear combination of the other weight vectors.

$$\mathbf{u}_j = \sum_{k \in [1,m] \setminus \{j\}} \lambda_k \mathbf{u}_k \tag{6.12}$$

Second, it assumes that the neuron activation function $\sigma(\cdot)$ is linear. Starting with Eqs. 6.9 and 6.10 we can write $\mathbb{E}_{\mathbf{x}}(y_l - y'_l)$ as

$$\begin{aligned}
& \mathbb{E}_{\mathbf{x}}\Big(\sigma(\mathbf{x}\mathbf{u}_j^T)V_{j,l} + \underbrace{\sum_{k \in [1,m] \setminus \{j\}} \sigma(\mathbf{x}\mathbf{u}_k^T)(V_{k,l} - V'_{k,l})}_{:=R}\Big) \\
\overset{\text{Eq. 6.12}}{=} \quad & \mathbb{E}_{\mathbf{x}}\Big(\sigma(\mathbf{x}(\sum_{k \in [1,m] \setminus \{j\}} \lambda_k \mathbf{u}_k)^T)V_{j,l} + R\Big) \\
\overset{\sigma(\cdot)\text{ lin.}}{=} \quad & \mathbb{E}_{\mathbf{x}}\Big(\sum_{k \in [1,m] \setminus \{j\}} \sigma(\mathbf{x}\mathbf{u}_k^T)\lambda_k V_{j,l} + R\Big) \\
= \quad & \sum_{k \in [1,m] \setminus \{j\}} \mathbb{E}_{\mathbf{x}}\Big(\sigma(\mathbf{x}\mathbf{u}_k^T)\Big)(V_{k,l} - V'_{k,l} + \lambda_k V_{j,l})
\end{aligned} \tag{6.13}$$

We set this term to zero (and thus satisfy Eq. 6.11) by setting each component of the sum to zero.

$$\forall k \in [1,m] \setminus \{j\} : V'_{k,l} = V_{k,l} + \lambda_k V_{j,l} \tag{6.14}$$

This condition is directly implemented by the update rule in our shrinking algorithm (Eq. 6.5 and 6.6).

Data-bound Shrinking Data-bound shrinking does not require linearity in $\sigma(\cdot)$. It rather assumes that the expected value of the neuron activity j is a linear combination of the expected values of the other activities:

$$\mathbb{E}_{\mathbf{x}}(\sigma(\mathbf{x}\mathbf{u}_j^T)) = \sum_{k \in [1,m] \setminus \{j\}} \lambda_k \mathbb{E}_{\mathbf{x}}(\sigma(\mathbf{x}\mathbf{u}_k^T)) \tag{6.15}$$

$\mathbb{E}_{\mathbf{x}}(\cdot)$ is estimated using importance sampling:

$$\hat{\mathbb{E}}_{\mathbf{x}}(\sigma(\mathbf{x}\mathbf{u}_k^T); \mathscr{X}) = \frac{1}{|\mathscr{X}|} \sum_{\mathbf{x}' \in \mathscr{X}} \sigma(\mathbf{x}'\mathbf{u}_k^T) \tag{6.16}$$

In practice, the samples in $\mathscr{X}$ are collected in the activity matrix A from Sec. 6.3.2. We can satisfy Eq. 6.11 by using the λ-values from Eq. 6.15, so that $\mathbb{E}_{\mathbf{x}}(y_l - y'_l)$ becomes

$$\begin{aligned}
\mathbb{E}_{\mathbf{x}}(y_l - y'_l) &\overset{\text{Eqs. 6.9,6.10}}{=} \mathbb{E}_{\mathbf{x}}\Big(\sigma(\mathbf{x}\mathbf{u}_j^T)V_{j,l} + \sum_{k \in [1,m] \setminus \{j\}} \sigma(\mathbf{x}\mathbf{u}_k^T)(V_{k,l} - V'_{k,l})\Big) \\
&= \mathbb{E}_{\mathbf{x}}(\sigma(\mathbf{x}\mathbf{u}_j^T)V_{j,l}) + \sum_{k \in [1,m] \setminus \{j\}} \mathbb{E}_{\mathbf{x}}(\sigma(\mathbf{x}\mathbf{u}_k^T))(V_{k,l} - V'_{k,l}) \\
&\overset{\text{Eq. 6.15}}{=} \sum_{k \in [1,m] \setminus \{j\}} \mathbb{E}_{\mathbf{x}}(\sigma(\mathbf{x}\mathbf{u}_k^T))(V_{k,l} - V'_{k,l} + \lambda_k V_{j,l})
\end{aligned} \tag{6.17}$$

Again, we set this to zero using Eq. 6.14.

6.6 Conclusion

We have described a generic method for improving the decoding speed and BLEU score of single system NMT. Our approach involves unfolding an ensemble of multiple systems into a single large neural network and shrinking this network by removing redundant neurons. We showed in Sec. 6.5 that our different shrinking methods set the expected error introduced by shrinking to zero under different assumptions. Our best results on Japanese-English either yield a gain of 2.2 BLEU compared to the original single NMT network at about the same decoding speed, or a 3.4× CPU decoding speed up with only a minor drop in BLEU.

The current formulation of unfolding works for networks of the same topology as the concatenation of layers is only possible for analogous layers in different networks. Unfolding and shrinking diverse networks could be possible, for example by applying the technique only to the input and output layers or by some other scheme of finding associations between units in different models, but we leave this investigation to future work as models in NMT ensembles in current research usually have the same topology.

We applied unfolding and shrinking to recurrent NMT architectures (Sec. 3.6.3). Our techniques could also be applied to more recent non-recurrent architectures such as the Transformer (Sec. 3.6.5) or ConvS2S (Sec. 3.6.4), but these architectures often already achieve sufficient decoding speed as they require less sequential computation and are thus more easily parallelizable.

My work consists of two parts: of the one which is here, and of everything which I have not written. And precisely this second part is the important one.

Ludwig Wittgenstein

7

The Role of Language Models in Neural Sequence-to-Sequence Prediction

This chapter draws from several publications, but with a special emphasis on the language modelling aspect. Sec. 7.2 is based on Hasler et al. (2017b), Sec. 7.3 on Stahlberg et al. (2019a); Stahlberg and Byrne (2019a), Sec. 7.4 on Stahlberg et al. (2018a), and Sec. 7.5 on Stahlberg et al. (2019b). Some text passages are verbatim copies from my own contributions to these publications.

7.1 Motivation

Language models (LMs) play a central role in traditional statistical machine translation (Ch. 2). For example, the synchronous grammar in hierarchical SMT like Hiero (Chiang, 2007) spans a vast space of possible translations, but is weak in assigning scores to them. The language model is mainly responsible for selecting a fluent and coherent translation from that space. However, language models have lost their key role in translation with the shift to the neural machine translation paradigm (Ch. 3). In contrast to generative models that use language model probabilities as priors, 'vanilla NMT' as introduced in Sec. 3.5 is a discriminative approach that is not amenable to integrating an LM. Language models are sometimes used in MT evaluation

systems in an ensemble with NMT (Bojar et al., 2018; Junczys-Dowmunt, 2018b; Stahlberg et al., 2019b; Wang et al., 2017e), but gains are generally small and less justifiable outside evaluation conditions. Most production-level NMT deployments do not use LMs at all (Wu et al., 2016b).

Outside translation, however, language models are still often a vital part of NLP systems. Here, we demonstrate the usefulness of language models in two related areas: word reordering (Sec. 7.2) and grammatical error correction (Sec. 7.3). We then return to machine translation and show in Sec. 7.4 how LMs can be used in NMT training. Our technique is effective in leveraging monolingual data and yields gains even on top of back-translation, especially in low-resource scenarios. In Sec. 3.17.4 we apply document-level language models to make our translation system sensitive to the context outside the current sentence. Cross-sentence context can be helpful to resolve ambiguity such as pronoun agreement or consistency in lexical choice.

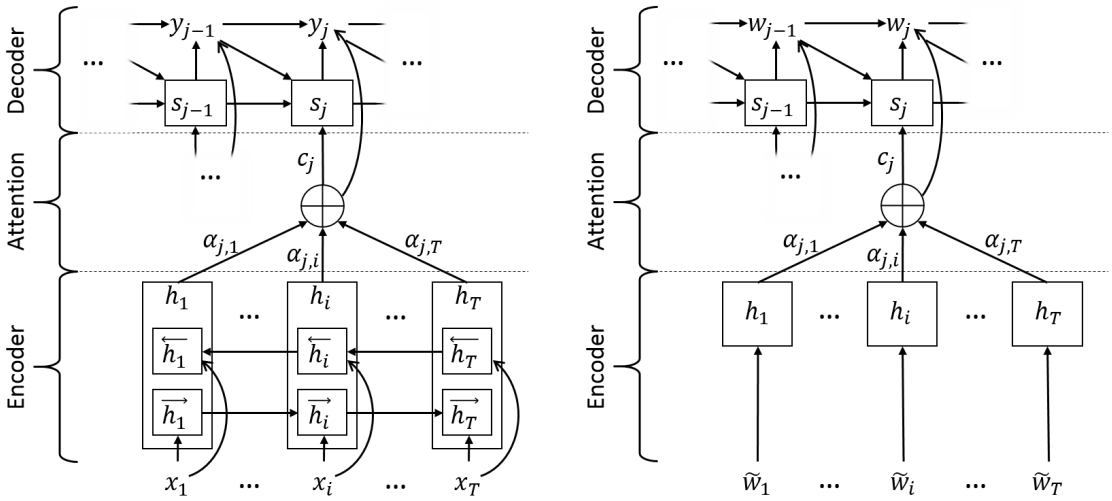
7.2 Neural Word Reordering Using Language Models

Word reordering as an isolated task aims to reorder a bag of words to form a fluent sentence. While this problem in that exact form rarely occurs in practical natural language processing, it is a useful proxy to study the capabilities of different models and search strategies as it requires an understanding of high-level aspects of language such as "syntactic structure, selective restrictions, subcategorization, and discourse considerations" (Elman, 1990). Therefore, this section can be seen as a targeted investigation into fluency in NLP models.

Prior work usually addressed word reordering either as *syntactic linearization* using syntactic, part-of-speech, and dependency information (Liu et al., 2015; Puduppully et al., 2016; Zhang et al., 2012; Zhang and Clark, 2011, 2015), or *LM-based linearization* which aims to find word permutations with high language model scores (de Gispert et al., 2014; Hasler et al., 2017b; Schmaltz et al., 2016). In this section, we report on the work of Hasler et al. (2017b) in which we proposed a new non-syntactic neural architecture for word reordering and combined it with an informed beam search variation that makes use of heuristics that are tailored towards the reordering problem. Our system surpasses prior work both in speed and accuracy on the Penn Treebank.

7.2.1 Bag-to-sequence Modeling with Attentional Neural Networks

For convenience, we formalize a bag of words as sequence $\tilde{\mathbf{w}} = \langle \tilde{w}_1, \ldots, \tilde{w}_T \rangle$ in which words are sorted, e.g. alphabetically, so that we can refer to the t-th word in the bag, and repeated



(a) RNNsearch model (Bahdanau et al., 2015) as in Sec. 3.6.3, Fig. 3.11.

(b) Our bag2seq model which is a variation of RNNsearch with a non-recurrent encoder.

Fig. 7.1 Difference between RNNsearch and the bag2seq model.

words are distinct tokens. Similarly to the NMT left-to-right factorization (Eq. 3.1) we can describe the probability of a particular (ordered) sentence $\mathbf{w} = w_1^T$ as:[1]

$$P(\mathbf{w}|\tilde{\mathbf{w}}) = \prod_{t=1}^{T} P(w_t|w_1^{t-1}, \tilde{\mathbf{w}}). \tag{7.1}$$

Schmaltz et al. (2016) modelled the conditionals $P(w_t|w_1^{t-1}, \tilde{\mathbf{w}})$ with a recurrent neural network language model (RNNLM) which implies the assumption that the probability is independent of the bag $\tilde{\mathbf{w}}$ ($P(w_t|w_1^{t-1}, \tilde{\mathbf{w}}) = P(w_t|w_1^{t-1})$). Unlike the RNNLM, our bag2seq model does not make that independence assumption. Bag2seq is derived from the recurrent NMT model RNNsearch (Sec. 3.6.3), but removes the recurrent connections in the encoder. Fig. 7.1 illustrates the difference between both architectures. The decoder is connected with the encoder via attention. As we pointed out in Sec. 3.6.1, attention does not have the concept of positions by itself and views the annotations (or *keys*) h_i as unordered set. The attention layer in bag2seq attends to the items in a bag of words $\tilde{\mathbf{w}}$ as it does not use recurrent connections or positional embeddings.

[1] In this section, we do not have a clear *source sentence* $\mathbf{x}$ and a *target sentence* $\mathbf{y}$ like in the rest of this thesis. We chose to use the special notation $\mathbf{w}$ to emphasize this fact.

7.2.2 Search

We have seen throughout this thesis that beam search is the ubiquitous decoding algorithm for neural sequence models. We have shown in Sec. 5.4.4 that beam search is prone to search errors. Applying beam search to word reordering is even more critical: Schmaltz et al. (2016) reported that gains often do not saturate even with a large beam of 512. They suggested adding the unigram probabilities of the remaining words in the bag as future cost estimates to the beam-search scoring function and reported large gains for an n-gram LM and an RNNLM. We re-implement this future cost heuristic, $h(\cdot)$, and further propose a new search heuristic, $g(\cdot)$, which collects unigram statistics during decoding. We keep hypotheses in the beam if their score is close to a theoretical upper bound which is the product of the best word probabilities given any history within the explored search space. For each word $\tilde{w} \in \tilde{\mathbf{w}}$ we maintain a heuristic score estimate $\hat{P}(\tilde{w})$ which we initialize to 0. Each time the search algorithm visits a new context, we update the estimates such that $\hat{P}(\tilde{w})$ is the current best score for $\tilde{w}$:

$$\hat{P}(\tilde{w}) = \max_{c \in \mathscr{C}_t} P(\tilde{w}|c, \tilde{\mathbf{w}}) \tag{7.2}$$

where $\mathscr{C}_t$ is the set of contexts (i.e. ordered prefixes in the form of w_1^t) explored by beam search so far. Thus, instead of computing a future cost, we compare the actual score of a partial hypothesis with the product of heuristic estimates of its words. This is especially useful for model combinations since all models are taken into account. More formally, at each time step t our beam search keeps the n best hypotheses according to scoring function $S(\cdot)$ using partial model score $f(\cdot)$ and estimates $g(\cdot)$:

$$\begin{aligned} S(w_1^t) &= f(w_1^t) - g(w_1^t) \\ f(w_1^t) &= \log P(w_1^t|\tilde{\mathbf{w}}) \\ g(w_1^t) &= \sum_{w' \in w_1^t} \log \hat{P}(w'). \end{aligned} \tag{7.3}$$

We recover standard beam search with $g \equiv 0$. In addition, we implement hypothesis recombination to further reduce the number of search errors.

7.2.3 Results

Hasler et al. (2017b) used the bag2seq model in a bilingual setting to improve translation on WMT English-German, and in a monolingual setting on the Penn Treebank (Marcus et al., 1993, PTB). In this section, we only report the results on the Penn Treebank as it is more focused on word reordering as an isolated task. We use pre-processed data by Schmaltz et al.

Model	dev			test		
	none	$h(\cdot)$	$g(\cdot)$	*none*	$h(\cdot)$	$g(\cdot)$
Previous work						
Gyro[2]	–	–	–	42.2	–	–
NGRAM-512	35.8	38.7	–	–	38.6	–
LSTM-512	38.0	42.5	–	–	42.7	–
LSTM-512+GW	–	43.7	–	–	44.5	–
Hasler et al. (2017b)						
n-gram	35.9	38.8	**39.1**	35.7	38.6	**38.9**
n-gram+GW	39.4	41.9	**43.1**	38.6	41.9	**42.7**
RNNLM	38.8	42.8	**43.6**	38.6	43.2	**44.2**
RNNLM+GW	44.3	49.5	**51.4**	44.1	49.9	**51.9**

Table 7.1 BLEU scores for PTB word-ordering for different search heuristics with beam=512. NGRAM-512, LSTM-512 are quoted from Schmaltz et al. (2016), and +GW indicates Gigaword data. The best results for a given model and set are highlighted in bold.

(2016) (ca. 40k sentences for training). Following Schmaltz et al. (2016), we chose a word-level vocabulary size of 16K with two different UNK tokens. We use the "medium" setup of Zaremba et al. (2014) for our RNNLM and a medium-sized bag2seq model (300-dimensional word embeddings, 500-dimensional LSTM (Hochreiter and Schmidhuber, 1997) cells). We also compare with the "Gyro" system of de Gispert et al. (2014) that tackles word reordering with a modified hierarchical phrase-based SMT system. Like prior work, we compute case-insensitive BLEU scores with the `mteval-v13.pl` script. The 5-gram count-based language models are estimated with the SRILM toolkit (Stolcke, 2002).

Tab. 7.1 summarizes the impact of the search heuristics under different models. Our *n*-gram and RNNLM models reproduce the results of Schmaltz et al. (2016) under *none* and $h(\cdot)$. Using our novel search heuristic $g(\cdot)$ yields significant gains across the board for all our language models. Interestingly, Tab. 7.2 shows that search heuristics are less important under the bag2seq model. Word reordering under bag2seq without heuristics outperforms the language models by a large margin (33.4 vs. 23.3/24.5 BLEU). Our best system is an ensemble of the RNNLM and the bag2seq model which also benefits from large beam sizes.

To sum up, we have compared several models for word reordering: two different kinds of LMs (*n*-gram and RNNLM) and a novel bag2seq model. Language model based word reordering benefits from our search heuristics, while the more sophisticated bag2seq model performs well even with simpler search.

[2]Gyro has an advantage because longer sentences are processed in chunks of maximum length 20 due to technical limitations.

Model	*none*	$h(\cdot)$	$g(\cdot)$
beam=5			
n-gram	23.3	**30.1**	26.5
RNNLM	24.5	**33.6**	29.7
bag2seq	**33.4**	27.0	31.7
RNNLM-ensemble	25.5	**34.2**	30.6
bag2seq-ensemble	34.8	**35.1**	32.8
RNNLM+bag2seq	35.7	**37.9**	34.4
RNNLM+bag2seq+*n*-gram	35.8	**38.2**	35.2
beam=10			
RNNLM-ensemble	29.2	**38.0**	36.7
RNNLM+bag2seq	37.5	**39.9**	38.7
beam=64			
RNNLM-ensemble	35.4	42.4	**43.2**
RNNLM+bag2seq	40.5	43.1	**43.5**

Table 7.2 Impact of search heuristics and beam sizes under different model combinations (test) on PTB data. The best results for a given setup are highlighted in bold.

7.3 Neural Grammatical Error Correction Using Finite State Transducers

Grammatical error correction (GEC) is the task of automatically correcting all types of errors in text; e.g. [*In a such situaction* → *In such a situation*]. Using neural models for GEC is becoming increasingly popular (Chollampatt and Ng, 2018; Ge et al., 2018a,b; Ji et al., 2017; Sakaguchi et al., 2017; Schmaltz et al., 2017; Xie et al., 2016; Yuan and Briscoe, 2016), possibly combined with phrase-based SMT (Chollampatt and Ng, 2017; Chollampatt et al., 2016; Grundkiewicz and Junczys-Dowmunt, 2018). A potential challenge for purely neural GEC models is their vast output space since neural seq2seq models such as discussed in Ch. 3 allow any kind of reorderings, replacements, insertions, and deletions. GEC is, compared to machine translation, a highly constrained problem as corrections tend to be very local, and lexical choices are usually limited. Finite state transducers (FSTs) are an efficient way to represent large structured search spaces (Sec. 2.6). In this section, we propose to construct a hypothesis space using standard FST operations like composition, and then constrain the output of a neural GEC system to that space. We study two different scenarios: In the first scenario, we do not have access to annotated training data, and only use a small development set for tuning. In this scenario, we construct the hypothesis space using word-level context-

independent confusion sets (Bryant and Briscoe, 2018) based on spell checkers and morphology databases, and rescore it with count-based and neural language models (NLMs). In the second scenario, we assume to have enough training data available to train SMT and neural machine translation (NMT) systems. In this case, we make additional use of the SMT lattice and rescore with an NLM-NMT ensemble.

7.3.1 Constructing the Hypothesis Space

Constructing the set of hypotheses The core idea of our approach is to first construct a (weighted) hypothesis space H which is large enough to be likely to contain good corrections, but constrained enough to embrace the highly structured nature of GEC. Then we use H to constrain a neural beam decoder. We make extensive use of the FST operations available in OpenFST (Allauzen et al., 2007) like composition (denoted with the $\circ$-operator) and projection (denoted with $\Pi_{\text{input}}(\cdot)$ and $\Pi_{\text{output}}(\cdot)$) to build H, following the notation introduced in Sec. 2.6.2. The process starts with an input lattice I. In our experiments without annotated training data, I is an FST which simply maps the input sentence to itself as shown in Fig. 7.2a. If we do have access to enough annotated data, we train an SMT system on it and derive I from the SMT n-best list.[3] For each hypothesis $\mathbf{y}$ we compute the Levenshtein distance $\text{lev}(\mathbf{x},\mathbf{y})$ to the source sentence $\mathbf{x}$. We construct a string $\mathbf{z}$ by prepending $\text{lev}(\mathbf{x},\mathbf{y})$ many $<\texttt{mcorr}>$ tokens to $\mathbf{y}$, and construct I such that:

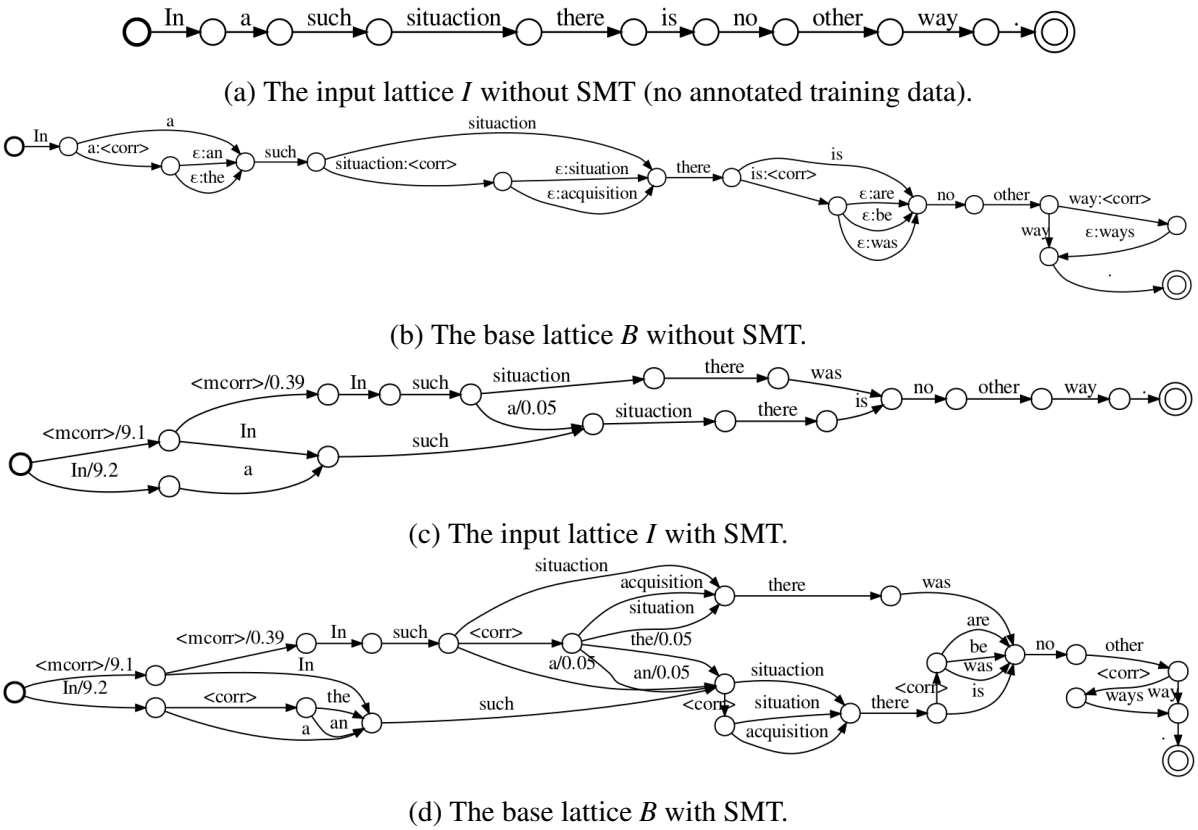
$$\mathbf{z} = (<\texttt{mcorr}>)^{\text{lev}(\mathbf{x},\mathbf{y})} \cdot \mathbf{y} \tag{7.4}$$

$$[[I]](\mathbf{z}) = -\lambda_{\text{SMT}}\text{SMT}(\mathbf{y}|\mathbf{x}). \tag{7.5}$$

We adapt the notation from Sec. 2.6 and denote the cost I assigns to mapping a string $\mathbf{z}$ to itself as $[[I]](\mathbf{z})$, and set $[[I]](\mathbf{z}) = \infty$ if I does not accept $\mathbf{z}$. $\text{SMT}(\mathbf{y}|\mathbf{x})$ is the SMT score. In other words, I represents the weighted SMT n-best list after adding $\text{lev}(\mathbf{x},\mathbf{y})$ many $<\texttt{mcorr}>$ tokens to each hypothesis as illustrated in Fig. 7.2c. We scale SMT scores by a factor λ_{SMT} for tuning.

Bryant and Briscoe (2018) addressed substitution errors such as non-words, morphology-, article-, and preposition-errors by creating confusion sets $C(x_i)$ that contain possible (context-independent) 1:1 corrections for each input word x_i. Specifically, they relied on CyHunspell for spell checking (Rodriguez and Seal, 2014), the AGID morphology database for morphology errors (Atkinson, 2011), and manually defined confusion sets for determiner and preposition errors, hence avoiding the need for annotated training data. We use the same confusion sets

[3] In the rare cases in which the n-best list did not contain the source sentence $\mathbf{x}$ we added it in a postprocessing step.



(a) The input lattice I without SMT (no annotated training data).

(b) The base lattice B without SMT.

(c) The input lattice I with SMT.

(d) The base lattice B with SMT.

Fig. 7.2 Building the hypothesis space for the input sentence "In a such situaction there is no other way .".

as Bryant and Briscoe (2018) to augment our hypothesis space via the edit flower transducer E shown in Fig. 7.3. E can map any sequence to itself via its σ-self-loop. Additionally, it allows the mapping $x_i \rightarrow <\texttt{corr}> \cdot\, y$ for each $y \in C(x_i)$. For example, for the misspelled word $x_i =$ 'situaction' and the confusion set $C(\text{'situaction'}) = \{\text{'situation'}, \text{'acquisition'}\}$, E allows mapping 'situaction' to '$<\texttt{corr}>$ situation' and '$<\texttt{corr}>$ acquisition', and to itself via the σ-self-loop. The additional $<\texttt{corr}>$ token will help us to keep track of the edits. We obtain our *base lattice* B which defines the set of possible hypotheses by composition and projection:

$$B := \Pi_{\text{output}}(I \circ E). \tag{7.6}$$

Fig. 7.2d shows B for our running example.

Scoring the hypothesis space We apply multiple scoring strategies to the hypotheses in B. First, we penalize $<\texttt{mcorr}>$ and $<\texttt{corr}>$ tokens with two further parameters, λ_{mcorr}

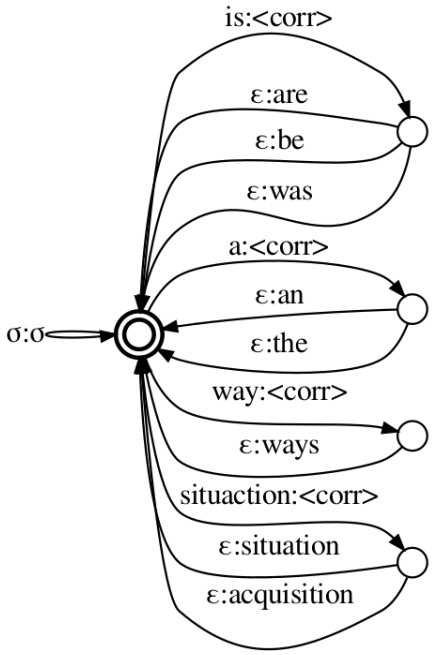


Fig. 7.3 The edit transducer E. The σ-label can match any input symbol.

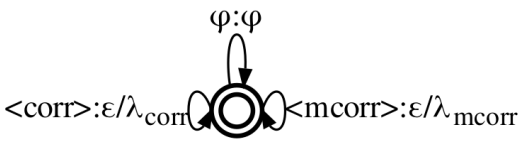


Fig. 7.4 The penalization transducer P. The ϕ-label can match any input except $<$ `corr` $>$ and $<$ `mcorr` $>$.

and λ_{corr}, by composing B with the penalization transducer P shown in Fig. 7.4.[4] The λ_{mcorr} and λ_{corr} parameters control the trade-off between the number and quality of the proposed corrections since high values bias towards fewer corrections.

To incorporate word-level language model scores we train a 5-gram count-based LM with KenLM (Heafield, 2011; Heafield et al., 2013) on the One Billion Word Benchmark dataset (Chelba et al., 2014), and convert it to an FST L using the OpenGrm NGram Library (Roark et al., 2012). For tuning purposes we scale weights in L with λ_{KenLM}:

$$[[L]](\mathbf{y}) = -\lambda_{\text{KenLM}} \log P_{\text{KenLM}}(\mathbf{y}). \tag{7.7}$$

[4] Rather than using $<$ `mcorr` $>$ and $<$ `corr` $>$ tokens and the transducer P we could directly incorporate the costs in the transducers I and E, respectively. We chose to use explicit correction tokens for clarity.

Our combined word-level scores can be expressed with the following transducer:

$$H_{\text{word}} = B \circ P \circ L. \tag{7.8}$$

Since we operate in the tropical semiring, path scores in H_{word} are linear combinations of correction penalties, LM scores, and, if applicable, SMT scores, weighted with the λ-parameters. Note that exact inference in H_{word} is possible using FST shortest path search. This is an improvement over the work of Bryant and Briscoe (2018) who selected correction options greedily. Our ultimate goal, however, is to rescore H_{word} with neural models such as an NLM and – if annotated training data is available – an NMT model. Since our neural models use subword units (Sennrich et al., 2016c, BPEs), we compose H_{word} with a transducer T which maps word sequences to BPE sequences. Our final transducer H_{BPE} which we use to constrain the neural beam decoder can be written as:

$$\begin{aligned} H_{\text{BPE}} &= \Pi_{\text{output}}(H_{\text{word}} \circ T) \\ &= \Pi_{\text{output}}(I \circ E \circ P \circ L \circ T). \end{aligned} \tag{7.9}$$

To help downstream beam decoding we apply ε-removal, determinization, minimization, and weight pushing (Mohri, 1997; Mohri and Riley, 2001) to H_{BPE}. We search for the best hypothesis $\mathbf{y}^*_{\text{BPE}}$ with beam search using a combined score of word-level symbolic models (represented by H_{BPE}) and subword unit based neural models:

$$\begin{aligned} \mathbf{y}^*_{\text{BPE}} = \arg\max_{\mathbf{y}_{\text{BPE}}} \Big(& - [[H_{\text{BPE}}]](\mathbf{y}_{\text{BPE}}) \\ & + \lambda_{\text{NLM}} \log P_{\text{NLM}}(\mathbf{y}_{\text{BPE}}) \\ & + \lambda_{\text{NMT}} \log P_{\text{NMT}}(\mathbf{y}_{\text{BPE}} | \mathbf{x}_{\text{BPE}}) \Big) \end{aligned} \tag{7.10}$$

Since H_{BPE} is determinized and does not have ε-arcs, if a string $\mathbf{y}_{\text{BPE}}$ is accepted by H_{BPE}, there is a unique path p for each prefix length $j \in [1, |\mathbf{y}_{\text{BPE}}|]$ which is labelled with the prefix $\mathbf{y}_{\text{BPE}}{}_1^j$. We denote the weight of the last arc in p (labelled with $\mathbf{y}_{\text{BPE}_j}$) with $[[H_{\text{BPE}}]]_{\text{part}}(\mathbf{y}_{\text{BPE}_j} | \mathbf{y}_{\text{BPE}}{}_1^{j-1})$. Eq. 7.10 can be rewritten as:

$$\begin{aligned} \mathbf{y}^*_{\text{BPE}} = \arg\max_{\mathbf{y}_{\text{BPE}}} \sum_{j=1}^{|\mathbf{y}_{\text{BPE}}|} \Big(& - [[H_{\text{BPE}}]]_{\text{part}}(\mathbf{y}_{\text{BPE}_j} | \mathbf{y}_{\text{BPE}}{}_1^{j-1}) \\ & + \lambda_{\text{NLM}} \log P_{\text{NLM}}(\mathbf{y}_{\text{BPE}_j} | \mathbf{y}_{\text{BPE}}{}_1^{j-1}) \\ & + \lambda_{\text{NMT}} \log P_{\text{NMT}}(\mathbf{y}_{\text{BPE}_j} | \mathbf{y}_{\text{BPE}}{}_1^{j-1}, \mathbf{x}_{\text{BPE}}) \Big) \end{aligned}$$

	Uses E	5-gram FST-LM	NLM (BPE)	CoNLL-2014 P	CoNLL-2014 R	CoNLL-2014 M2	CoNLL-2014 GLEU	JFLEG Test P	JFLEG Test R	JFLEG Test M2	JFLEG Test GLEU
1	Bryant and Briscoe (2018)			40.56	20.81	34.09	59.35	76.23	28.48	57.08	48.75
2	✓	✓		40.62	20.72	34.08	64.03	81.08	28.69	59.38	48.95
3	✓	✓	✓	54.43	25.21	44.19	66.75	79.88	32.99	62.20	50.93
4	✓	✓	✓	53.64	26.34	44.43	66.89	70.24	38.94	60.51	52.61

Table 7.3 Results without using annotated training data. Systems are tuned with respect to the metric highlighted in gray. Input lattices I are derived from the source sentence as in Fig. 7.2a.

This form lends itself to beam search since hypotheses can be built up from left to right. This final decoding pass can be seen as an ensemble of a neural LM and an NMT model which is constrained at each time step by the set of possible tokens in H_{BPE}. This is similar to our hybrid SMT-NMT scheme presented in Sec. 4.3.

We have introduced three λ-parameters λ_{corr}, λ_{KenLM}, and λ_{NLM}, and three additional parameters λ_{SMT}, λ_{mcorr}, and λ_{NMT} if we make use of annotated training data. We also use a word insertion penalty λ_{wc} for our SMT-based experiments. We tune all these parameters on the development sets using our tuning algorithm from Sec. 5.6.[5]

7.3.2 Experiments

Experimental setup In our experiments with annotated training data we use the SMT system of Junczys-Dowmunt and Grundkiewicz (2016)[6] to create 1000-best lists from which we derive the input lattices I. All our LMs are trained on the One Billion Word Benchmark dataset (Chelba et al., 2014). Our neural LM is a Transformer decoder architecture in the `transformer_base` configuration trained with Tensor2Tensor (Vaswani et al., 2018). Our NMT model is a Transformer model (`transformer_base`) trained on the concatenation of the NUCLE corpus (Dahlmeier et al., 2013) and the Lang-8 Corpus of Learner English v1.0 (Mizumoto et al., 2012). We only keep sentences with at least one correction (659K sentences in total). Both NMT and NLM models use byte pair encoding (Sennrich et al., 2016c, BPE) with 32K merge operations. We delay SGD updates by 2 on four physical GPUs as suggested by Saunders et al. (2018). We decode with beam size 12 using the SGNMT decoder (Ch. 5). We evaluate on CoNLL-2014 (Ng et al., 2014) and JFLEG-Test (Napoles et al., 2017), using CoNLL-2013 (Ng et al., 2013) and JFLEG-Dev as development sets. Our evaluation metrics are GLEU (Napoles et al., 2015) and M2 (Dahlmeier and Ng, 2012). We generated M2 files

[5] Similarly to Bryant and Briscoe (2018), even in our experiments without annotated *training* data, we do need a very small amount of annotated sentences for tuning.

[6] https://github.com/grammatical/baselines-emnlp2016

CoNLL-2014

	Uses E	5-gram FST-LM	NMT (BPE)	NLM (BPE)	P	R	M2	GLEU
1	Grundkiewicz and Junczys-Dowmunt (2018)				66.77	34.49	56.25	n/a
2	Unconstrained single NMT				54.98	22.20	42.45	67.19
3					60.95	26.21	48.18	68.30
4	✓	✓			57.58	32.39	49.83	68.82
5			✓	✓	65.26	33.03	54.61	69.92
6	✓		✓	✓	64.55	37.33	56.33	70.30
7	✓		✓(4x)	✓	66.71	38.97	58.40	70.60
8	✓		✓(4x)	✓	66.96	38.62	58.39	70.60

JFLEG Test

	Uses E	5-gram FST-LM	NMT (BPE)	NLM (BPE)	P	R	M2	GLEU
1	Grundkiewicz and Junczys-Dowmunt (2018)				n/a	n/a	n/a	61.50
2	Unconstrained single NMT				67.49	38.47	58.64	50.71
3					66.64	40.68	59.09	50.86
4	✓	✓			71.60	42.45	62.95	53.20
5			✓	✓	76.35	40.55	64.89	51.75
6	✓		✓	✓	78.85	47.72	69.75	55.39
7	✓		✓(4x)	✓	82.15	47.82	71.84	55.60
8	✓		✓(4x)	✓	74.19	56.41	69.79	58.63

Table 7.4 Results with using annotated training data. Systems are tuned with respect to the metric highlighted in gray. Input lattices I are derived from the Moses 1000-best list as in Fig. 7.2c. Row 3 is the SMT baseline.

using ERRANT (Bryant et al., 2017) for JFLEG and Tab. 7.3 to be comparable to Bryant and Briscoe (2018), but used the official M2 files in Tab. 7.4 to be comparable to Grundkiewicz and Junczys-Dowmunt (2018).

Results Our LM-based GEC results without using annotated training data are summarized in Tab. 7.3. Even when we use the same resources (same LM and same confusion sets) as Bryant and Briscoe (2018), we see gains on JFLEG (rows 1 vs. 2), probably because we avoid search errors in our FST-based scheme. Adding an NLM yields significant gains across the board. Tab. 7.4 shows that adding confusion sets to SMT lattices is effective even without neural models (rows 3 vs. 4). Rescoring with neural models also benefits from the confusion sets (rows 5 vs. 6). With our ensemble systems (rows 7 and 8) we are able to outperform prior

	Grundkiewicz and Junczys-Dowmunt (2018)		This work	
	CoNLL (M2)	JFLEG (GLEU)	CoNLL (M2)	JFLEG (GLEU)
SMT	50.27	55.79	48.18	50.86
Hybrid	56.25	61.50	58.40	58.63
Rel. gain	**11.90%**	**10.23%**	**21.21%**	**15.28%**

Table 7.5 Improvements over SMT baselines.

Hypothesis space	Error rate
Expanded input sentence (Tab. 7.3)	61.28%
SMT lattice (Tab. 7.4, rows 3, 5)	55.64%
Expanded SMT lattice (Tab. 7.4, rows 4, 6-8)	48.17%

Table 7.6 Oracle error rates for different hypothesis spaces using the first annotator in CoNLL-2014.

work[7] (row 1) on CoNLL-2014 and come within 3 GLEU on JFLEG. Since the baseline SMT systems of Grundkiewicz and Junczys-Dowmunt (2018) were better than the ones we used, we achieve even higher relative gains over the respective SMT baselines (Tab. 7.5).

Oracle experiments Our FST-based composition cascade is designed to enrich the search space to allow the neural models to find better hypotheses. Tab. 7.6 reports the oracle sentence error rate for different configurations, i.e. the fraction of reference sentences in the test set which are not in the FSTs. Expanding the SMT lattice significantly reduces the oracle error rate from 55.63% to 48.17%.

In conclusion, we demonstrated that our FST-based approach to GEC outperforms prior work on LM-based GEC significantly, especially when combined with a neural LM. We also applied our approach to SMT lattices and reported much better relative gains over the SMT baselines than previous work on hybrid systems. Our results suggest that FSTs provide a powerful and effective framework for constraining neural GEC systems.

[7] We compare our systems to the work of Grundkiewicz and Junczys-Dowmunt (2018) as they also studied hybrid systems and used similar training data. We note, however, that Ge et al. (2018b) reported even better results with much more (non-public) training data. Comparing (Ge et al., 2018a) and (Ge et al., 2018b) suggests that most of their gains come from the larger training set. Very recently, Zhao et al. (2019) have set a new best M2 score on CoNLL-2014 of 61.15%.

Fig. 7.5 Deletion FST D which can map any token in the list R from Tab. 7.7 to ε. The σ-label matches any symbol and maps it to itself.

Deletion Frequency (BEA dev set)	Token
164	the
78	,
50	a
33	to
20	it
18	of
16	in
12	that
8	will
8	have
8	for
8	an
7	is
7	-
6	they
6	's
6	and
5	had

Table 7.7 List of tokens R that can be deleted by the deletion transducer D in Fig. 7.5.

7.3.3 UCAM@BEA19: FST-based GEC in Shared Task Submissions

The University of Cambridge participated in the BEA-2019[8] shared task on grammatical error correction with three submissions: (a) a submission to the low-resource track that extended the LM-based system from Tab. 7.3, (b) a purely neural system, and (c) a joint submission with the Cambridge University Computer Laboratory that used our FST framework for system combination. In this section, we focus on how our FST pipeline was used for these submissions, and refer to Stahlberg and Byrne (2019a) and Yuan et al. (2019) for a more detailed description of the full submission.

For our shared task submissions we enhanced the FST cascade presented in the previous sections as follows:

1. We compose the input I with the deletion transducer D in Fig. 7.5. D allows to delete tokens on the short list shown in Tab. 7.7 at a cost λ_{del}. We selected R by looking up all

[8] 14th Workshop on Innovative Use of NLP for Building Educational Applications

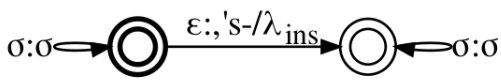


Fig. 7.6 Insertion FST A for adding the symbols ",", "-", and "'s" at a cost of λ_{ins}. The σ-label matches any symbol and maps it to itself at no cost.

Sub	Del	Ins	NLM	Beam	CoNLL-2014			BEA-2019 Dev		
					P	R	M2	P	R	ERRANT
Tab. 7.3 (Stahlberg et al., 2019a)					54.12	25.52	44.21	n/a		
✓			1x	8	58.59	24.14	45.58	42.44	14.68	30.79
✓	✓		1x	8	59.01	26.07	47.11	41.21	16.47	31.69
✓	✓	✓	1x	8	52.89	26.68	44.20	40.09	19.97	33.36
✓	✓	✓	2x	8	54.05	26.71	44.87	40.70	20.01	33.73
✓	✓	✓	2x	16	57.05	27.22	46.80	42.02	19.76	34.29
✓	✓	✓	2x	32	58.48	28.21	48.15	42.37	19.92	34.58

Table 7.8 Results on the low-resource track. The λ-parameters are tuned on the BEA-2019 dev set.

tokens which have been deleted in the dev set more than five times and manually filtered that list slightly. We did not use the full list of dev set deletions to avoid under-estimating λ_{del} in tuning.

2. In addition to the confusion sets used previously, we extracted all substitution errors from the BEA-2019 dev set which occurred more than five times, and added a small number of manually defined rules that fix tokenization around punctuation symbols.

3. We found it challenging to allow insertions in LM-based GEC because the LM often prefers inserting words with high unigram probability such as articles and prepositions more than less predictable words like proper names. We therefore restrict insertions to the three tokens ",", "-", and "'s" and allow only one insertion per sentence. We achieve this by adding the transducer A in Fig. 7.6 to our composition cascade.

Results without annotated training data Tab. 7.8 summarizes our low-resource experiments. On BEA-2019 Dev we report ERRANT (Bryant et al., 2017) scores to be comparable with the official evaluation results. Our substitution-only system already outperforms our system from the previous section (Stahlberg et al., 2019a). Allowing for deletions and insertions improves the ERRANT score on BEA-2019 Dev by 2.57 points. We report further gains on both test sets by ensembling two language models and increasing the beam size.

	Per Sentence		Per Word	
	CoNLL-2014	BEA-2019 Dev	CoNLL-2014	BEA-2019 Dev
Missing	0.35	0.46	1.51%	2.30%
Replacement	1.52	1.31	6.62%	6.57%
Unnecessary	0.42	0.19	1.83%	0.98%
Total	2.29	1.96	9.95%	9.86%

Table 7.9 Number of correction types in CoNLL-2014 and BEA-2019 Dev references.

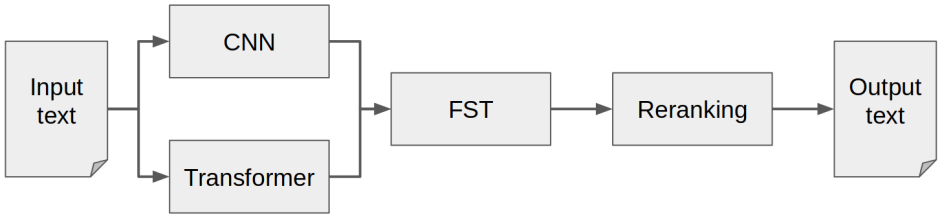


Fig. 7.7 Pipeline of the system combination following Yuan et al. (2019).

	P	R	ERRANT
CNN	41.72	22.46	35.61
Transformer	53.17	32.90	47.34
FST	57.33	32.38	49.68
Reranking	57.64	33.53	50.39

Table 7.10 ERRANT scores on BEA-2019 Dev for the components in Fig. 7.7.

Our results in Tab. 7.8 differ significantly between the CoNLL-2014 test set and the BEA-2019 dev set. Allowing insertions is beneficial on BEA-2019 Dev but decreases the M2 score on CoNLL-2014. Increasing the beam size improves our system by 3.28 points on CoNLL-2014 while the impact on BEA-2019 Dev is smaller (+0.85 points). These differences can be partially explained by comparing the error type frequencies in the reference annotations in both test sets (Tab. 7.9). Samples in CoNLL-2014 generally need more corrections per sentence than in BEA-2019 Dev. More importantly, the CoNLL-2014 test set contains fewer missing words, but much more unnecessary words than BEA-2019 Dev. This mismatch tempers with tuning as we explicitly tune insertion and deletion penalties.

Results with annotated training data Our FST framework was also used in a system combination entry together with the Cambridge University Computer Laboratory (Yuan et al., 2019). The system pipeline is shown in Fig. 7.7. 'CNN' (Yuan et al., 2019) and 'Transformer' (Stahlberg and Byrne, 2019a) are two purely neural system that generate n-best lists

for the FST stage. The union of both n-best lists take the role of the Moses SMT system in Sec. 7.3.2. The output of the FST component is reranked following Yannakoudakis et al. (2017) to produce the final output. Tab. 7.10 shows that each stage of the processing pipeline improves the accuracy of the grammatical error correction system. For more details we refer to Yuan et al. (2019).

7.4 Simple Fusion: Using Language Models for NMT Training

In the previous two sections we explored the use of language models for word reordering and grammatical error correction. We will now return to machine translation and present a method which makes use of language models to improve NMT training.

Machine translation relies on parallel training data, which is difficult to acquire. In contrast, monolingual data is abundant for many languages and domains. Traditional statistical machine translation (SMT) effectively leverages monolingual data using language models (LMs) (Brants et al., 2007). As outlined in Sec. 2.3, the combination of LM and translation model (TM) in SMT can be traced back to the noisy-channel model which applies the Bayes rule to decompose a translation system. In contrast, NMT uses a discriminative model and learns the distribution $P(\mathbf{y}|\mathbf{x})$ directly end-to-end. Therefore, the vanilla training regimen for NMT is not amenable to integrating an LM or monolingual data in a straightforward manner. We have discussed several approaches that improve NMT with monolingual data in Sec. 3.9. Some of these techniques (e.g. *shallow fusion* and *deep fusion* (Gulcehre et al., 2015, 2017b)) use language models, but integrate them only at inference time. In this section we present a novel technique called *simple fusion* (Stahlberg et al., 2018a) that integrates the LM scores during NMT training. Our training procedure first trains an LM on a large monolingual corpus. We then hold the LM fixed and train the NMT system to optimize the combined score of LM and NMT on the parallel training set. This allows the NMT model to focus on modeling the source sentence, while the LM handles the generation based on the target-side history. Sriram et al. (2018) explored a similar idea for speech recognition using a gating network for controlling the relative contribution of the LM. We show that our simpler architecture without an explicit control mechanism is effective for machine translation. We observe gains of up to more than 2 BLEU points from adding the LM to TM training. We also show that our method can be combined with back-translation (Sennrich et al., 2016b), yielding further gains over systems without LM.

7.4.1 Translation Model Training under Language Model Predictions

Shallow fusion (Gulcehre et al., 2015) integrates an LM by changing the decoding objective to:

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} \log P_{\text{TM}}(\mathbf{y}|\mathbf{x}) + \lambda \log P_{\text{LM}}(\mathbf{y}). \tag{7.11}$$

$P_{\text{LM}}(\cdot)$ is produced by an LSTM-based RNN-LM (Mikolov et al., 2010) which has been trained on monolingual target language data. $P_{\text{TM}}(\cdot)$ can be a typical encoder-decoder NMT model. λ is a hyper-parameter which is tuned on the development set.

Shallow fusion combines a fixed TM with a fixed LM at inference time. Sriram et al. (2018) proposed to keep the LM fixed, but train a sequence-to-sequence (Seq2Seq) NMT model from scratch which includes the LM as a fixed part of the network. They argue that this approach allows the Seq2Seq network to use its model capacity for the conditioning on the source sequence since the language modeling aspect is already covered by the LM. Their *cold fusion* architecture includes a gating network which learns to regulate the contributions of the LM at each time step. They demonstrated superior performance of cold fusion on a speech recognition task.

In spirit of the cold fusion technique of Sriram et al. (2018) we also keep the LM fixed when training the translation network. However, we greatly simplify the architecture by removing the need for a gating network. We follow the usual left-to-right factorization in NMT (Eq. 3.1):

$$P(\mathbf{y}|\mathbf{x}) = \prod_{j=1}^{|\mathbf{y}|} P(y_j|y_1^{j-1}, \mathbf{x}). \tag{7.12}$$

Let $S_{\text{TM}}(y_j|y_1^{j-1}, \mathbf{x})$ be the output of the TM projection layer without softmax, i.e., what we would normally call the logits. We investigate two different ways to parameterize $P(y_j|y_1^{j-1}, \mathbf{x})$ using $S_{\text{TM}}(y_j|y_1^{j-1}, \mathbf{x})$ and a fixed and pre-trained language model $P_{\text{LM}}(\cdot)$: POSTNORM and PRENORM.

POSTNORM This variant is directly inspired by shallow fusion as we turn $S_{\text{TM}}(y_j|y_1^{j-1}, \mathbf{x})$ into a probability distribution using a softmax layer, and sum its log-probabilities with the log-probabilities of the LM, i.e. multiply their probabilities:

$$P(y_j|y_1^{j-1}, \mathbf{x}) = \text{softmax}(S_{\text{TM}}(y_j|y_1^{j-1}, \mathbf{x})) \cdot P_{\text{LM}}(y_j|y_1^{j-1}). \tag{7.13}$$

PRENORM Another option is to apply normalization after combining the raw $S_{\text{TM}}(y_j|y_1^{j-1},\mathbf{x})$ scores with the LM log-probability:

$$P(y_t|y_1^{t-1},\mathbf{x}) = \text{softmax}\Big(S_{\text{TM}}(y_t|y_1^{t-1},\mathbf{x}) + \log P_{\text{LM}}(y_t|y_1^{t-1})\Big). \tag{7.14}$$

Discussion of POSTNORM **and** PRENORM

Note that $P(y_j|y_1^{j-1},\mathbf{x})$ might not represent a valid probability distribution under the POSTNORM criterion since, as component-wise product of two distributions, it is not guaranteed to sum to 1. A way to fix this issue would be to combine TM and LM probabilities in the probability space rather than in the log space. However, we have found that probability space combination does not work as well as POSTNORM in our experiments. $S_{\text{TM}}(y_j|y_1^{j-1},\mathbf{x})$ under POSTNORM can be seen as the residual probability added to the prediction of the LM.

It is interesting to investigate what signal is actually propagated into $S_{\text{TM}}(y_j|y_1^{j-1},\mathbf{x})$ when training with the PRENORM strategy. We can rewrite $P(y_j|y_1^{j-1},\mathbf{x})$ as:

$$P(y_j|y_1^{j-1},\mathbf{x}) = \frac{P(y_j,y_1^{j-1}|\mathbf{x})}{P(y_1^{j-1}|\mathbf{x})} = \frac{P(y_j,\mathbf{x}|y_1^{j-1})}{P(\mathbf{x}|y_1^{j-1})} = \frac{P(\mathbf{x}|y_j,y_1^{j-1})}{P(\mathbf{x}|y_1^{j-1})} P(y_j|y_1^{j-1}). \tag{7.15}$$

Alternatively, we can decompose $P(y_j|y_1^{j-1},\mathbf{x})$ as follows using Eq. 7.14:

$$\begin{aligned} P(y_j|y_1^{j-1},\mathbf{x}) =& \text{softmax}\Big(S_{\text{TM}}(y_j|y_1^{j-1},\mathbf{x}) + \log P_{\text{LM}}(y_j|y_1^{j-1})\Big) \\ \propto& \exp\Big(S_{\text{TM}}(y_j|y_1^{j-1},\mathbf{x}) + \log P_{\text{LM}}(y_j|y_1^{j-1})\Big) \\ =& \exp(S_{\text{TM}}(y_j|y_1^{j-1},\mathbf{x})) \cdot P_{\text{LM}}(y_j|y_1^{j-1}). \end{aligned} \tag{7.16}$$

Combining Eq. 7.15 and Eq. 7.16 leads to:

$$\exp(S_{\text{TM}}(y_j|y_1^{j-1},\mathbf{x})) \propto \frac{P(\mathbf{x}|y_1^{j})}{P(\mathbf{x}|y_1^{j-1})} \tag{7.17}$$

This means that $S_{\text{TM}}(y_j|y_1^{j-1},\mathbf{x})$ under PRENORM is trained to predict how much more likely the *source* sentence becomes when a particular target token y_j is revealed.

7.4.2 Experimental Setup

We evaluate our method on a variety of publicly available and proprietary data sets. For our Turkish-English (tr-en), English-Turkish (en-tr), and Estonian-English (et-en) experiments we

Language pair	# Sentences
Turkish-English (WMT)	207.7K
Estonian-English (WMT)	2,178.0K
Xhosa-English (INTERNAL)	739.2K

Table 7.11 Parallel training data.

Language	# Sentences	LM Perplexity	
		dev	test
English (WMT)	26.9M	91.16	87.77
Turkish (WMT)	3.0M	59.19	70.46
English (INTERNAL)	20.0M	105.28	108.19

Table 7.12 Monolingual training data.

use all available parallel data from the WMT18 evaluation campaign to train the translation models. Our language models are trained on *News Crawl 2017*. We use *news-test2017* as development ("dev") set and *news-test2018* as test set.

Additionally, we use a proprietary corpus of public posts on Facebook. We refer to it as 'INTERNAL' data set. This corpus consists of monolingual English in-domain sentences and parallel data in Xhosa-English. Training set sizes are summarized in Tables 7.11 and 7.12.

Our preprocessing consists of lower-casing, tokenization, and subword-segmentation using joint byte pair encoding (Sennrich et al., 2016c) with 16K merge operations. On Turkish, we additionally remove diacritics from the text.

On WMT we use lower-cased SacreBLEU[9] (Post, 2018) to be comparable with the literature.[10] On our internal data we report tokenized BLEU scores.

Our Seq2Seq models are recursive encoder-decoder architectures as introduced in Sec. 3.6.3 with dot-product attention (Luong et al., 2015b) trained with the PyTorch Translate library.[11] Both decoder and encoder consist of two 512-dimensional LSTM layers and 256-dimensional embeddings. The first encoder layer is bidirectional, the second one runs from right to left. Our training and architecture hyperparameters are summarized in Tab. 7.13. Our LSTM-based LMs have the same size and architecture as the decoder networks, but do not use attention and do not condition on the source sentence. We run beam search with beam size of 6 in all our experiments.

[9] SacreBLEU signature for tr-en test-2017:
BLEU+c.lc+l.tr-en+#.1+s.exp+t.wmt17+tok.13a+v.1.2.10

[10] For translation into Turkish we evaluate after diacritics removal.

[11] https://github.com/pytorch/translate

Architecture hyperparameters	
Source vocab size (BPE)	16,000
Target vocab size (BPE)	16,000
Embedding size (all)	256
Encoder LSTM units	512
Encoder layers	2
Decoder LSTM units	512
Decoder layers	2
Attention type	dot product
Training settings	
Optimization	Vanilla SGD
Learning rate	0.5
Batch size	32
Label smoothing ε	0.1
Checkpoint averaging	Last 10

Table 7.13 Summary of NMT settings for all models in this section.

For each setup we train five models using SGD (batch size of 32 sentences) with learning rate decay and label smoothing, and either select the best one (single system) or ensemble the four best models based on dev set BLEU score.

7.4.3 Results

Tab. 7.14 compares our methods PRENORM and POSTNORM on the tested language pairs. Shallow fusion often leads to minor improvements over the baseline for both single systems and ensembles. We also reimplemented the *cold fusion* technique for comparison. For our machine translation experiments we report mixed results with cold fusion, with performance ranging between 0.33 BLEU gain on Xhosa-English and slight BLEU degradation in most of our Turkish-English experiments.

Both of our methods, PRENORM and POSTNORM yield significant improvements in BLEU across the board. We report more consistent gains with POSTNORM than with PRENORM. All our POSTNORM systems outperform both shallow fusion and cold fusion on all language pairs, yielding test set gains of up to +2.36 BLEU (Xhosa-English ensembles).

7.4.4 Discussion and Analysis

Back-translation As described in Sec. 3.9, a very popular technique to use monolingual data for NMT is back-translation (Sennrich et al., 2016b). Back-translation uses a reverse NMT

English-Turkish (WMT)

Method	Single		4-Ensemble	
	dev	test	dev	test
Baseline (no LM)	12.23	11.56	14.17	13.35
Shallow fusion	12.45	11.61	14.43	13.51
Cold fusion	12.39	11.54	14.20	13.23
PRENORM	12.82	11.93	**14.78**	13.41
POSTNORM	**13.30**	**12.27**	14.77	**13.61**

Turkish-English (WMT)

Method	Single		4-Ensemble	
	dev	test	dev	test
Baseline (no LM)	16.14	16.60	18.01	18.67
Shallow fusion	16.11	16.70	18.01	18.67
Cold fusion	16.25	16.21	17.99	18.40
PRENORM	15.88	16.39	17.95	18.40
POSTNORM	**16.59**	**17.03**	**18.38**	**19.17**

Estonian-English (WMT)

Method	Single		4-Ensemble	
	dev	test	dev	test
Baseline (no LM)	16.02	16.57	16.83	17.91
Shallow fusion	16.02	16.57	16.83	17.91
Cold fusion	15.40	15.99	16.48	17.79
PRENORM	**16.80**	**17.44**	**17.78**	**19.01**
POSTNORM	16.43	17.10	17.62	18.63

Xhosa-English (INTERNAL)

Method	Single		4-Ensemble	
	dev	test	dev	test
Baseline (no LM)	10.39	11.49	13.87	15.43
Shallow fusion	10.69	11.65	14.06	15.54
Cold fusion	10.72	11.29	13.66	15.13
PRENORM	11.06	12.13	14.50	16.07
POSTNORM	**12.34**	**13.27**	**15.45**	**17.79**

Table 7.14 Comparison of our PRENORM and POSTNORM combination strategies with shallow fusion (Gulcehre et al., 2015) and cold fusion (Sriram et al., 2018) under a recurrent neural network language model (RNN-LM).

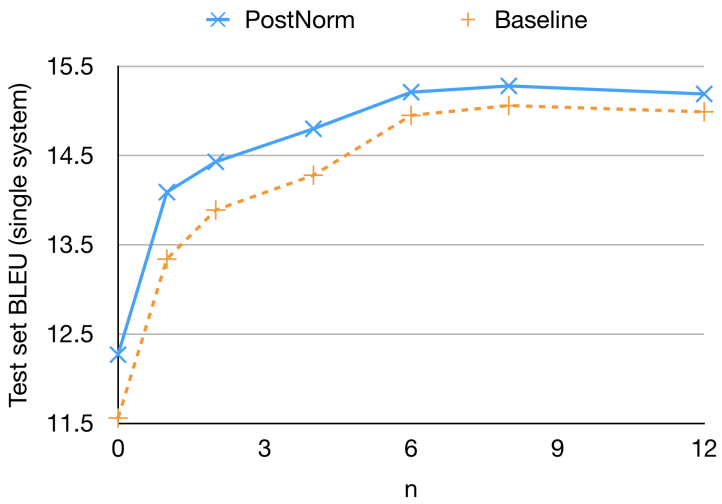


Fig. 7.8 Performance using back-translation on English-Turkish. Synthetic sentences are mixed at a ratio of 1:n where n is plotted on the x-axis.

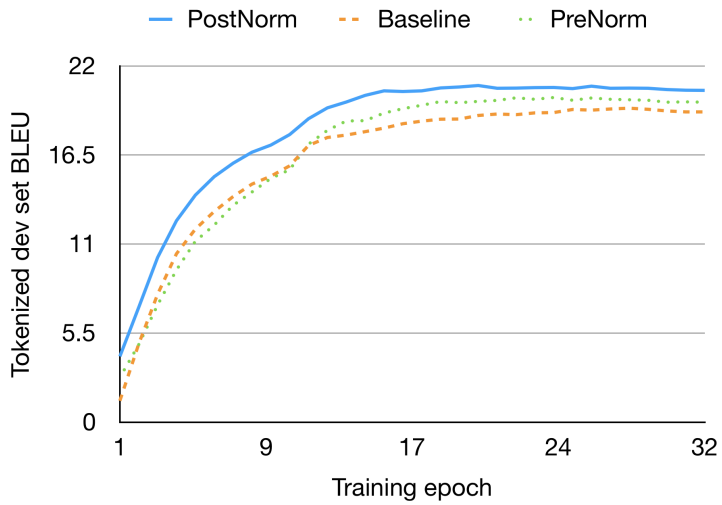


Fig. 7.9 Convergence of NMT training with and without LM on English-Turkish.

system to translate monolingual target language sentences into the source language, and adds the newly generated sentence pairs to the training data. The amount of monolingual data which can be used for back-translation is usually limited by the size of the parallel corpus as the translation quality suffers when the mixing ratio between synthetic and real source sentences is too large (Poncelas et al., 2018). This is a severe limitation particularly for low-resource MT. Fig. 7.8 shows that both our baseline system without LM and our POSTNORM system benefit greatly from back-translation up to a mixing ratio of 1:8, but degrade slightly if this ratio is exceeded. POSTNORM is significantly better than the baseline even when using it in combination with back-translation.

English-Turkish (WMT, single system)

Method	Dev set		Test set	
	FFN	RNN	FFN	RNN
Baseline (no LM)	12.23		11.56	
Shallow fusion	12.25	12.45	11.53	11.61
Cold fusion	12.33	12.39	11.51	11.54
PRENORM	**12.76**	12.82	**11.82**	11.93
POSTNORM	12.65	**13.30**	11.79	**12.27**

Table 7.15 Comparison between using a recurrent LM (RNN) and an n-gram based feedforward LM (FFN) on English-Turkish.

English-Turkish (WMT), POSTNORM strategy

LM type		Single		4-Ensemble	
FFN	RNN	dev	test	dev	test
		12.23	11.56	14.17	13.35
✓		12.65	11.79	14.36	13.48
	✓	13.30	12.27	14.77	13.61
✓	✓	12.86	12.02	14.72	13.70

Table 7.16 Combining an RNN-LM and a feedforward LM with the translation model using the POSTNORM strategy.

Training convergence We have found that training converges faster under the POSTNORM loss. Fig. 7.9 plots the training curves of our systems. The baseline (orange curve) reaches its maximum of 19.39 BLEU after 28 training epochs. POSTNORM surpasses this BLEU score already after 12 epochs.

Language model type So far we have used recurrent neural network language models (Mikolov et al., 2010, RNN-LM) with LSTM cells in all our experiments. We can also parameterize an n-gram language model with a feedforward neural network (Bengio et al., 2003, FFN-LM). In order to compare both language model types we trained a 4-gram feedforward LM with two 512-dimensional hidden layers and 256-dimensional embeddings on Turkish monolingual data. Tab. 7.15 shows that the PRENORM strategy works particularly well for the n-gram LM. However, using an RNN-LM with the POSTNORM strategy still gives the best overall performance. Using both RNN and n-gram LM at the same time does not improve translation quality any further (Tab. 7.16).

Method	Perplexity	Average entropy
Baseline (no LM)	23.46	3.19
RNN-LM	59.19	4.66
TM under POSTNORM	113.69	1.82

Table 7.17 Perplexity and average entropies of the distributions generated by our systems on the English-Turkish dev set.

Impact on the TM distribution With the POSTNORM strategy, the TM still produces a distribution over the target vocabulary as the scores are normalized before the combination with the LM. This raises a natural question: How different are the distributions generated by a TM trained under POSTNORM loss from the distributions of the baseline system without LM? Tab. 7.17 gives some insight to that question. As expected, the RNN-LM has higher perplexity than the baseline as it is a weaker model of translation. The RNN-LM also has a higher average entropy which indicates that the LM distributions are smoother than those from the baseline translation model. The TM trained under POSTNORM loss has a much higher perplexity which suggests that it strongly relies on the LM predictions and performs poorly when it is not combined with it. However, the average entropy is much lower (1.82) than both other models, i.e. it produces much sharper distributions.

Language models improve fluency A traditional interpretation of the role of an LM in MT is that it is (also) responsible for the fluency of translations (Koehn, 2010). Thus, we would expect more fluent translations from our method than from a system without LM. Tab. 7.18 breaks down the BLEU score of the baseline and the PRENORM ensembles on Estonian-English into n-gram precisions. Most of the BLEU gains can be attributed to the increase in precision of higher order n-grams, indicating improvements in fluency. Tab. 7.19 shows some examples where our PRENORM system produces a more fluent translation than the baseline.

Method	BLEU	Precisions				BP
		1-gram	2-gram	3-gram	4-gram	
Baseline (no LM)	17.91	53.0	23.7	12.3	6.6	0.996
PRENORM	19.01	54.0	24.9	13.4	7.4	1.000
Relative improvement	+6.14%	**+1.89%**	**+5.06%**	**+8.94%**	**+12.12%**	–

Table 7.18 BLEU n-gram precisions for Estonian-English.

Source	Eestis ja Hispaanias peeti kinni neli Kemerovo grupeeringu liiget
Reference	Four members of the Kemerovo group arrested in Estonia and Spain
Baseline (no LM)	In Estonia and Spain, four kemerovo groups were held
PRENORM	Four Kemerovo group members were held in Estonia and Spain
Source	Ta ütleb, et elab aastaid hiljem endiselt hirmus.
Reference	He says that years later, he still lives in fear.
Baseline (no LM)	He says that, for years, he still lives in fear.
PRENORM	He says that many years later he still lives in fear.
Source	“Ma kardan," ütleb ta.
Reference	“I’m afraid," he says.
Baseline (no LM)	“I fear," says he.
PRENORM	“I am afraid," he says.

Table 7.19 Translation samples from the Estonian-English test set.

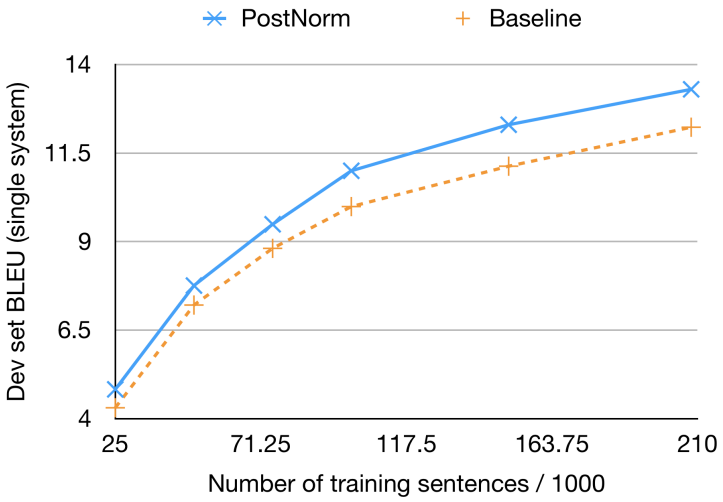


Fig. 7.10 English-Turkish BLEU over training set size.

Training set size We artificially reduced the size of the English-Turkish training set even further to investigate how well our method performs in low-resource settings (Fig. 7.10). Our POSTNORM strategy outperforms the baseline regardless of the number of training sentences, but the gains are smaller on very small training sets.

We have presented a simple yet very effective method to use language models in NMT which incorporates the LM already into NMT training. We reported significant and consistent gains from using our method in four language directions over two alternative ways to integrate LMs into NMT (*shallow fusion* and *cold fusion*) and showed that our approach works well even in combination with back-translation and on top of ensembles. Our method leads to faster training convergence and more fluent translations than a baseline system without LM.

7.5 UCAM@WMT19: Document-level Language Models for Translation

Two techniques provided the fabric of the Cambridge University Engineering Department's entry to the WMT19 evaluation campaign: elastic weight consolidation (EWC) and different forms of language modelling (LMs). In this section we focus on the language modelling aspect of the submission and refer to Stahlberg et al. (2019b) for a full discussion of the system.

Inspired by the shallow fusion technique by Gulcehre et al. (2015, 2017b) we ensemble our neural translation models with neural language models. While this technique is effective for single models, the gains are diminishing under NMT ensembles trained with large amounts of back-translated sentences. To incorporate document-level context in a light-weight fashion, we propose a modification to the Transformer (Vaswani et al., 2017) that has separate attention layers for inter- and intra-sentential context. We report large perplexity reductions compared to sentence-level LMs under the new architecture. Our document-level LM yields small BLEU gains on top of strong NMT ensembles, and we hope to benefit even more from it in document-level human evaluation.

Furthermore, in this section we view MBR-based NMT (Sec. 4.5) as building a specialized n-gram LM for each sentence that computes the risk of hypotheses relative to SMT lattices. We show that even though the performance gap between NMT and traditional statistical machine translation (SMT) is growing rapidly on the task at hand, SMT can still improve very strong NMT ensembles.

7.5.1 Document-level Language Modelling

MT systems usually translate sentences in isolation. However, there is evidence that humans also take context into account, and judge translations from humans with access to the full document higher than the output of a state-of-the-art sentence-level machine translation system (Läubli et al., 2018). Common examples of ambiguity which can be resolved with cross-sentence context are pronoun agreement or consistency in lexical choice. The WMT19 competition encouraged submissions of translation systems that are sensitive to cross-sentence context. We explored the use of document-level language models to enhance a sentence-level translation system. We argue that this is a particularly light-weight way of incorporating document-level context. First, the LM can be trained independently on monolingual target language documents, i.e. no parallel or source language documents are needed. Second, since our document-level decoder operates on the n-best lists from a sentence-level translation system, existing translation infrastructure does not have to be changed – we just add another

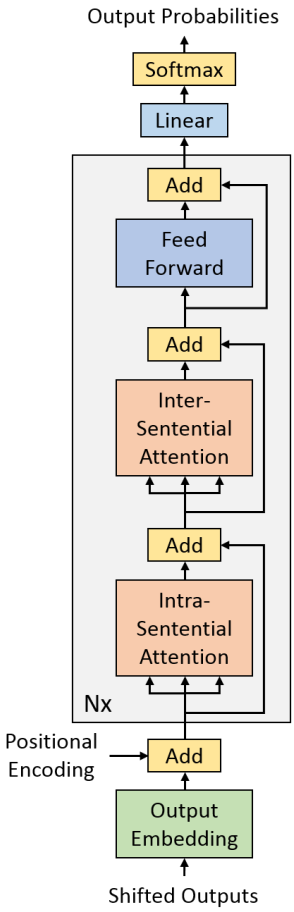


Fig. 7.11 Our modified Intra-Inter Transformer architecture with two separate attention layers.

(document-level) decoding pass. On a practical note, this means that, by skipping the second decoding pass, our system would work well even for the translation of isolated sentences when no document context is available.

Our document-level LMs are trained on the concatenations of all sentences in target language documents, separated by special sentence boundary tokens. Training a standard Transformer LM (Vaswani et al., 2017) on this data already yields significant reductions in perplexity compared to sentence-level LMs. However, the attention layers have to capture two kinds of dependencies – the long-range cross-sentence context and the short-range context within the sentence. Our modified Intra-Inter Transformer architecture (Fig. 7.11) splits these two responsibilities into two separate layers using masking. The "Intra-Sentential Attention" layer only allows to attend to the previous tokens in the current sentence, i.e. the intra-sentential attention mask activates the tokens between the most recent sentence boundary marker and the current symbol. The "Inter-Sentential Attention" layer is restricted to the tokens in all previous *complete* sentences, i.e. the mask enables all tokens from the document beginning to the most

Word	Position	Attention mask	
		Intra-sentential	Inter-sentential
Vinyl	0	0	1
destination	1	0	1
:	2	0	1
who	3	0	1
is	4	0	1
actually	5	0	1
buying	6	0	1
records	7	0	1
?	8	0	1
< /s >	9	0	1
Lonely	10	1	0
,	11	1	0
middle-aged	12	1	0
men	13	1	0
love	14	1	0
'???'	15	1	0

Fig. 7.12 Intra-sentential and inter-sentential attention masks for an English example from `news-test2017`. Document-level context helps to predict the next word ('vinyl').

recent sentence boundary marker. As usual (Vaswani et al., 2017), during training the attention masks are also designed to prevent attending to future tokens. Fig. 7.12 shows an example of the different masks. Note that as illustrated in Fig. 7.11, both attention layers are part of the same layer stack which allows a tight integration of both types of context. An implication of this design is that they also use the same positional embedding – the positional encoding for the first unmasked item for intra-sentential attention may not be zero. For example, 'Lonely' has the position 10 in Fig. 7.12 although it is the first word in the current sentence.

We use our document-level LMs to rerank n-best lists from a sentence-level translation system. Our initial document is the first-best sentence hypotheses. We greedily replace individual sentences with lower-ranked hypotheses (according to the translation score) to drive up a combination of translation and document LM scores. We start with the sentence with the minimum difference between the first- and second-best translation scores. We stop when the translation score difference to the first-best translation exceeds a threshold.

7.5.2 Results

Our experimental setup is essentially as described in Sec. 4.6 for our WMT18 submission in the previous year (Stahlberg et al., 2018b): Our pre-processing includes Moses tokenization,

Model	Context	Perplexity (per subword)							
		German				English			
		test15	test16	test17	test18	test15	test16	test17	test18
Standard	Sentence-level	36.23	35.69	36.17	34.77	39.94	37.19	35.34	42.38
Standard	Document-level	26.63	27.85	25.43	28.36	43.37	34.55	31.27	39.74
Intra-Inter	Document-level	23.54	22.39	22.05	22.56	34.25	31.16	29.31	34.47

Table 7.20 Language model perplexities of different neural language models. The standard model has 448M parameters, Intra-Inter has 549M parameters.

English-German

		Base	Big (with EWC)	
		Single	Single	4-Ensemble
1	Using back-translation?	No	Yes	Yes
2	NMT	43.8	47.8	48.8
3	+ Sentence-level LM	44.7	47.8	48.8
4	+ PBSMT (MBR-based)	45.1	48.0	49.1
5	+ Document-level Intra-Inter LM	45.7	47.6	49.3

German-English

		Base	Big (with EWC)	
		Single	Single	4-Ensemble
1	Using back-translation?	No	Yes	Yes
2	NMT	40.7	47.4	48.3
3	+ Sentence-level LM	41.4	47.6	48.3
4	+ PBSMT (MBR-based)	42.1	47.6	48.5
5	+ Document-level Intra-Inter LM	42.1	47.3	48.6

Table 7.21 Using different kinds of language models for translation on `news-test2018`. The Big models are fine-tuned on old WMT test sets with EWC as described by Stahlberg et al. (2019b). The PBSMT baseline gets 26.7 BLEU on English-German and 27.5 BLEU on German-English.

punctuation normalization, truecasing, and joint subword segmentation using byte pair encoding (Sennrich et al., 2016c) with 32K merge operations. We compute cased BLEU scores with `mteval-v13a.pl` that are directly comparable with the official WMT scores.[12] Our models are trained with the TensorFlow (Abadi et al., 2016) based Tensor2Tensor (Vaswani et al., 2018) library and decoded with our SGNMT framework (Ch. 5). We delay SGD updates (Saunders

[12] http://matrix.statmt.org/

English-German	
Team	BLEU
MSRA	44.9
Microsoft	43.9
NEU	43.5
UCAM	**43.0**
Facebook FAIR	42.7
JHU	42.5
eTranslation	41.9
8 more...	

German-English	
Team	BLEU
MSRA	42.8
Facebook FAIR	40.8
NEU	40.5
UCAM	**39.7**
RWTH	39.6
MLLP-UPV	39.3
DFKI	38.8
4 more...	

Table 7.22 English-German and German-English primary submissions to the WMT19 shared task.

et al., 2018) to use larger training batch sizes than our technical infrastructure[13] would normally allow with vanilla SGD by using the `MultistepAdam` optimizer in Tensor2Tensor. We use Transformer (Vaswani et al., 2017) 'Big' models for our final system. We use `news-test2017` as development set to tune model weights and select checkpoints and `news-test2018` as test set.

Tab. 7.20 shows that our document-level Intra-Inter architecture achieves much better perplexity than both a sentence-level language model and a document-level vanilla Transformer model. Tab. 7.21 summarizes our translation results with various kinds of language models. Adding a Transformer sentence-level LM to NMT helps for the single Base model without back-translation, but is less effective on top of (ensembles of) Big models with back-translation (row 2 vs. 3). We once again confirm the effectiveness of our MBR-based combination scheme for NMT-SMT hybrids in Sec. 4.5: Extracting n-gram probabilities from traditional PBSMT lattices and using them as source-conditioned n-gram LMs yields gains even on top of our ensembles (row 4). Our document-level Intra-Inter language models improve the ensembles and the single En-De Base model, but hurt performance slightly for the single Big models (row 5).

To sum up, our WMT19 submission focused on regularized fine-tuning and language modelling. In this section we focused on the language modeling aspect. With our novel Intra-Inter Transformer architecture for document-level LMs we achieved significant reductions in perplexity and minor improvements in BLEU over very strong baselines. Our systems are competitive on both English-German and German-English (Tab. 7.22), especially considering the immense speed with which our field has been advancing in recent years (Tab. 7.23).

[13]The Cambridge HPC service (http://www.hpc.cam.ac.uk/) allows parallel training on up to four physical P100 GPUs.

Year	Best in competition	Our submission	Δ
2017	28.3	32.8	**+4.5**
2018	48.3	49.3	**+1.0**
2019	44.9	43.0	**-1.9**

Table 7.23 Comparison of our English-German system with the winning submissions over the past two years.

7.6 Conclusion

In this chapter we have explored various ways to use language models for sequence-to-sequence prediction. In Sec. 7.2 we demonstrated that language models are useful to find the correct word order in a sentence when paired with sophisticated search procedures such as beam search with heuristic future cost estimates. Language models are also very useful for grammatical error correction (Sec. 7.3), especially when no annotated parallel data is available. Our approach involved constructing a search space for grammatical error correction using a cascade of FST operations such as introduced in Sec. 2.6.2, and rescoring the resulting FSTs with large neural language models. We devoted the remainder of the chapter to machine translation. Our *Simple Fusion* technique in Sec. 7.4 integrates language model predictions into NMT training and thus helps NMT to make better use of its model capacity. In Sec. 7.5 we used various kinds of language models for our submission to the WMT19 news translation shared task (Bojar et al., 2019), including a document-level language model with a novel Intra-Inter Transformer architecture that makes it possible to carry context across sentence boundaries.

Those who know nothing of foreign languages know nothing of their own.

Johann Wolfgang von Goethe

8

The Role of Hierarchical Models in Neural Sequence-to-Sequence Prediction

Some isolated paragraphs in the formal description in Sec. 8.3 overlap with my contributions to the work of Zhang et al. (2019a), but in this thesis we use a slightly different set of models and a completely different experimental evaluation. A shorter version of Sec. 8.4 has been published in conference proceedings (Stahlberg et al., 2018c) and some text passages are quoted verbatim.

8.1 Motivation

It is widely assumed that human language is hierarchical to a large extent – an idea that was promoted perhaps most prominently by Noam Chomsky in his work on formal grammars (Chomsky, 1957). In this chapter, we discuss approaches that incorporate different forms of hierarchy into neural sequence prediction. Closest to Chomsky's linguistic notion of hierarchy is Sec. 8.2 that integrates target-side syntactic parse trees into NMT models. However, hierarchical structure can be useful even if it does not correspond to phrase structure in the linguistic sense. For example, the production rules and non-terminal symbols in hierarchical statistical machine translation systems like Hiero (Chiang, 2007) are learned from data and

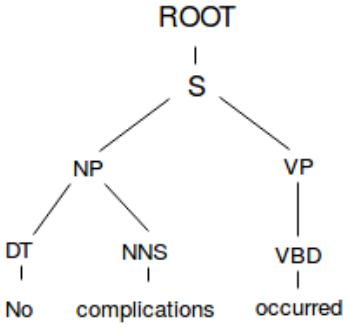
Representation	Sample
(a) Plain-text	No complications occurred
(b) Constituency tree	
(c) Linearized tree	(ROOT (S (NP (DT No) (NNS complications)) (VP (VBD occurred))))
(d) Derivation	ROOT→S ; S→NP VP ; NP→DT NNS ; DT→No ; NNS→complications ; VP→VBD ; VBD→occurred
(e) Linearized derivation	S</R> NP VP</R> DT NNS</R> No complications VBD</R> occurred
(f) POS/plain-text	DT No NNS complications VBD occurred

Table 8.1 Syntactic representations of the sentence "No complications occurred" following Saunders et al. (2018).

are not necessarily linguistically motivated (Sec. 2.7). Therefore, we have already shown in Ch. 4 on SMT-NMT hybrid systems how (non-linguistic) hierarchical structure (in the form of Hiero) can improve NMT. We will show in Sec. 8.3 that the combination of symbolic formal grammars and neural sequence models is also very useful for text normalization as it combines the predictive power of neural models with the guarantees on adequacy that come with symbolic models. Finally, in Sec. 8.4 we present our work on operation sequence neural machine translation (OSNMT). In OSNMT, the neural model predicts a sequence of operations/actions that generates the target sentence. OSNMT does not only have practical advantages as it yields hard alignment information, but it also has strong links to formal grammar theory as an operation sequence can be seen as a derivation in a multitext grammar.

8.2 Syntax-based NMT with Multi-representation Ensembles

In (computational) linguistics, the hierarchical structure of language is often represented as a tree such as the syntactic constituency tree in Tab. 8.1b. Using syntax trees for translation has been in focus of MT research since the early days of rule-based systems, with varying success. There is a solid body of research on syntax-based traditional statistical machine

External representation	Internal representation	Test BLEU
Linearized tree		28.4
Linearized derivation		28.7
POS/BPE		29.1
Plain BPE	Plain BPE	29.2
Linearized derivation	Linearized derivation	28.8
Linearized tree	Plain BPE	28.9
Plain BPE	Linearized derivation	28.8
Linearized derivation	Plain BPE	29.4
POS/BPE	Plain BPE	29.3
Plain BPE	POS/BPE	29.4

Table 8.2 Japanese-English syntax-based NMT with Transformer based models (Saunders et al., 2018). The first three rows are single models, the last five rows are multi-representation ensembles.

translation (Williams et al., 2016). Syntax has also been used for NMT (Sec. 3.17.2). This section contains a brief introduction to the work of Saunders et al. (2018) and summarizes some of the main results.

Saunders et al. (2018) studied the use of target-side syntax in NMT. Rather than changing the neural architecture, they changed the output representation of the target sentence. The target sentence is still represented as a linear sequence of tokens, but the token sequence conveys syntactic information. A straight-forward representation is the *linerarized* parse tree (Tab. 8.1c) based on bracket expressions (Aharoni and Goldberg, 2017b). Saunders et al. (2018) also proposed a representation that views the tree as sequence of rules (Tab. 8.1e) and which is shorter than linerarized trees. Finally, similarly to Nadejde et al. (2017), the plain text can be interspersed with POS tags (Tab. 8.1f).

Saunders et al. (2018) showed that ensembling different syntactic representations helps syntax-based NMT. The method heavily relies on the concept of SGNMT predictor wrappers introduced in Sec. 5.3.2: SGNMT decodes with an *external* representation using two predictors. The first predictor is a model which has been trained on the external representation. The second predictor is masked by the *parse* predictor wrapper that converts the external representation to a different, *internal*, representation. This is similar to the *fsttok* predictor wrapper discussed in Sec. 5.3.2 that defines the conversion between representations with an FST. Tab. 8.2 shows that multi-representation ensembles generally work better than single models or ensembles under the same representation. We refer to Saunders et al. (2018) for a more detailed description of the experimental setup and a deeper discussion of the results.

8.3 Neural Text Normalization Under Covering Grammars

Text normalization is a less studied yet very crucial part of many practical speech processing systems. For example, natural sounding speech synthesis depends on the correct pronunciation of numbers, and whether the string '123' should be spoken as 'one hundred twenty three' or 'one two three'. Text normalization is the task of converting the textual representation of numbers or other semiotic classes such as abbreviations to their spoken form while both conveying meaning and morphology.

Text normalization is a sequence-to-sequence prediction task (e.g. '1 2 3' $\rightarrow$ 'one twenty three'). However, it differs markedly from the problems we have discussed earlier in this thesis like machine translation and grammatical error correction for the following reasons (Zhang et al., 2019a):

1. While training data for machine translation can be crawled from manually translated websites and grammatical error correction systems can be trained on English learner corpora, data for text normalization must be curated and collected in dedicated efforts: While websites are often translated manually for a wider audience, there is little reason to e.g. write out the vocalization of a number.

2. The text normalization problem is highly context-dependent. Abbreviations and numbers often have different feasible vocalizations, and it depends on the context which one is appropriate. The problem is even more severe in languages like Russian in which the number words need to be inflected to be in agreement with context words.

3. The prediction problem is very unbalanced as the vast majority of tokens remain unchanged (i.e. mapped to the special token $<$ `self` $>$). This imbalance contributes to the data sparsity problem: large amounts of text only yield a small number of non-trivial text normalization samples.

4. The fault tolerance for a text normalization system is very low for so-called 'catastrophic' errors (see below).

Prior work (Sproat and Jaitly, 2016) has found that standard neural architectures for sequence-to-sequence tasks such as attention-based encoder-decoder networks with LSTM or GRU cells (discussed in Sec. 3.6.3) are able to yield very good overall accuracy when the input sequence is obtained with a sliding window over the segmented input sentence. However, these models are prone to occasional 'catastrophic errors' such as reading '11/10/2008' as 'the tenth of october two thousand eight' rather than 'the tenth of november two thousand eight' (Sproat and Jaitly, 2016) which are not acceptable in a production setting.

In this section, we follow Zhang et al. (2019a) and view text normalization as a *contextual* sequence-to-sequence problem. We discuss two contextual neural architectures for text normalization: `mutilevel_textnorm` and `ffcontext_textnorm`. `mutilevel_textnorm` consists of four GRU sub-networks: two for modelling the left and the right context, one for encoding the token to be normalized, and one for generating the normalized form. Each of these GRUs can run on different tokenization levels: Words or subword units (BPEs or word pieces) are suitable for modelling the left and right context. Subword units or characters are suitable to represent the token under consideration as they avoid out of vocabulary issues. The output/decoder GRU in `mutilevel_textnorm` is word-based as a small set of words is usually required to produce the normalized form. A small word-level vocabulary on the output side also ensures good compatibility with external symbolic models for incorporating constraints and downstream systems like TTS.

`mutilevel_textnorm` is able to condition on the entire sentence. However, we have found that modelling long range context is not always necessary for text normalization. Therefore, we also introduce a second architecture, the `ffcontext_textnorm` model, that uses a feedforward neural network to compute a context vector from a window around the to-be-normalized token.

Our models have clear speed advantages compared to the standard sequence-to-sequence model on sliding input windows while improving the overall accuracy slightly at the same time. Unfortunately, none of our neural models can prevent catastrophic errors. To eliminate the risk of such catastrophic errors while keeping the gains from using neural networks we follow Sproat and Jaitly (2016); Zhang et al. (2019a) and constrain the neural sequence models with formal covering grammars. We report substantial gains from using covering grammars on English for a number of different neural architectures, particularly in low resource settings.

8.3.1 Neural Text Normalization Models

The Baseline Model (`std_textnorm`)

We use the approach of Sproat and Jaitly (2016) as baseline. They use a standard RNNsearch model (Sec. 3.6.3) to map a sliding window around the to-be-normalized token to its normalized form. The token to be normalized is enclosed with special $<$`norm`$>\ldots<$`/norm`$>$ tags. For example the token *123* in the context *I live at ... King Ave .* would appear as

live at $<$`norm`$>$ 123 $<$`/norm`$>$ King Ave

on the input side with a context window size of 2, which would map to

one twenty three

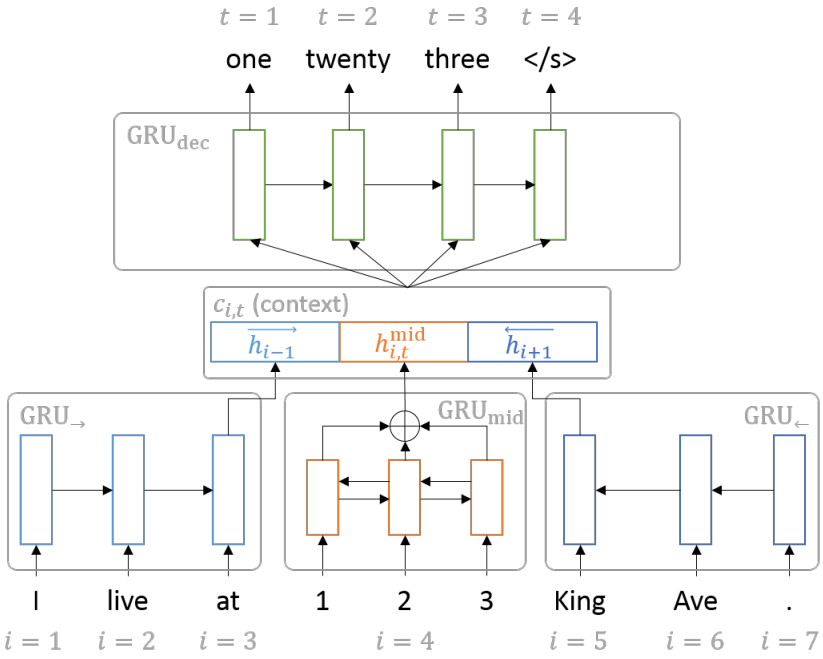


Fig. 8.1 The `multilevel_textnorm` model.

on the output side. The input sequence is segmented into characters, the output vocabulary consists of full words. Input tokens that remain unchanged are mapped to the special token $<$ `self` $>$. We will refer to this model as `std_textnorm` model.

Using Multiple Tokenization Levels (`multilevel_textnorm`)

Our notation in this section differs from the one introduced in Sec. 1.4, mainly because we need to include the context of the sequence-to-sequence mapping which may use a different tokenization scheme. We formalize a sentence as a sequence of pairs $(x_1, y_1), \ldots, (x_n, y_n)$ where n is the sentence length in segments. We denote the set of all segments as S and the output vocabulary as W. We assume pre-segmented data: each $x_i \in S$ is a single segment such as a complete date, address, money expression, etc (e.g. $x_i =$ "$1 million"), $y_i \in \{<$ `self` $>\} \cup W^+$ contains the normalized form of x_i as sequence of words (e.g. $y_i = \langle$"one", "million", "dollars"$\rangle$), or $<$ `self` $>$ if x_i does not require any normalization. The model uses two functions $\text{tok}_{\text{context}} : S \to T^+_{\text{context}}$ and $\text{tok}_{\text{mid}} : S \to T^+_{\text{mid}}$ which map segments to the token sequences for the context (T_{context}) and the word to normalize (T_{mid}). In most of our later experiments we use subword units for the context and characters for the middle word, i.e. T_{context} is the set of word pieces and T_{mid} the character set of the language.

The network architecture of the `multilevel_textnorm` model is illustrated in Fig. 8.1. We follow the general framework of encoder-decoder sequence-to-sequence models with an

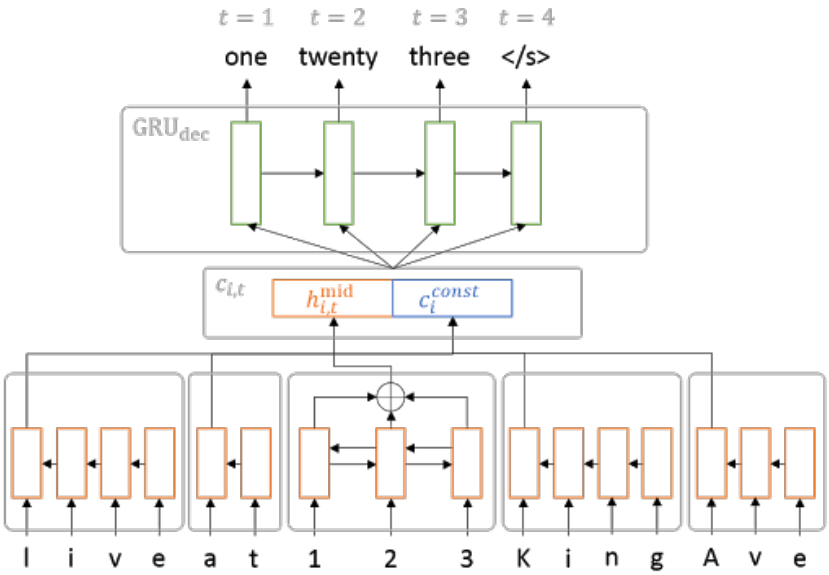


Fig. 8.2 The `ffcontext_textnorm` model.

encoder architecture which is specialized to text normalization. The decoder cell GRU_{dec} normalizes the i-th segment according the standard factorization for sequence models (Eq. 3.1):

$$P(y_i|x_i) = \prod_{t=1}^{|y_i|} P((y_i)_t|(y_i)_1^{t-1}, c_{i,t}) \tag{8.1}$$

where $c_{i,t}$ is a context vector for the t-th output token in segment i which is generated by the encoder network from the input tokens as concatenation of three vectors:

$$c_{i,t} = (\overrightarrow{h_{i-1}}; c_{i,t}^{\text{mid}}; \overleftarrow{h_{i+1}}). \tag{8.2}$$

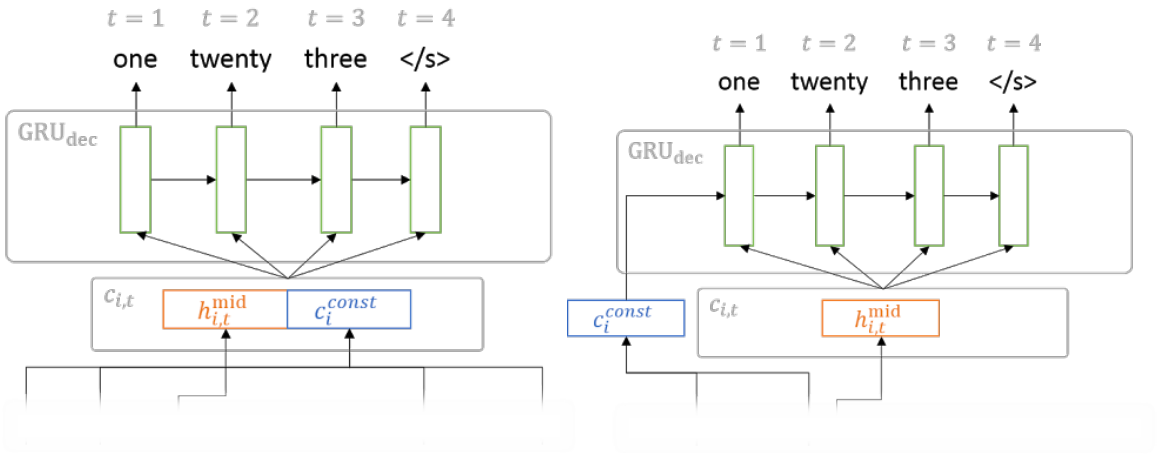
$\overrightarrow{h_i}$ and $\overleftarrow{h_i}$ are hidden states of a bidirectional GRU on the T_{context} tokenization level running over the complete input sequence $x_1, \ldots, x_n$:[1]

$$\overrightarrow{h_i} = \text{GRU}_{\rightarrow}(\overrightarrow{h_{i-1}}, \text{tok}_{\text{context}}(x_i)) \tag{8.3}$$

$$\overleftarrow{h_i} = \text{GRU}_{\leftarrow}(\overleftarrow{h_{i+1}}, \text{tok}_{\text{context}}(x_i)) \tag{8.4}$$

$c_{i,t}^{\text{mid}}$ is computed using attention over the hidden states of yet another bidirectional RNN GRU_{mid} which encodes the word to normalize by consuming the token sequence $\text{tok}_{\text{mid}}(x_i)$.

[1] Since $\text{tok}_{\text{context}}(\cdot)$ can yield multiple tokens, these equations may require multiple steps.



(a) The `concat` strategy.

(b) The `init` strategy.

Fig. 8.3 Using context vectors in the decoder network.

Context Modelling with Feedforward Networks (`ffcontext_textnorm`)

The `multilevel_textnorm` model uses RNNs ($\text{GRU}_{\rightarrow}$ and $\text{GRU}_{\leftarrow}$) to encode the full input sentence as context. However, long range context may not be always important in text normalization. Thus, the `ffcontext_textnorm` model uses a feedforward neural network which transforms the source annotations in a window around the to-be-normalized token to a context vector. Therefore, the only difference between `multilevel_textnorm` and `ffcontext_textnorm` is how $c_{i,t}$ is defined. The complete model is illustrated in Fig. 8.2. Rather than using the concatenation of hidden states of the context GRUs and $c^{\text{mid}}_{i,t}$, `ffcontext_textnorm` concatenates $c^{\text{mid}}_{i,t}$ with the vector c^{const}_i. c^{const}_i is produced by a feedforward neural network which takes the first backward hidden states of GRU_{mid} for each token in the context window around i.

Using Context in the Decoder Network (`concat` vs. `init`)

The previous sections concatenated (`concat`) a vector $c^{\text{mid}}_{i,t}$ with a constant context vector (c^{const}_i in `ffcontext_textnorm` or $(\overrightarrow{h_{i-1}}, \overleftarrow{h_{i+1}})$ in `multilevel_textnorm`) to obtain the input $c_{i,t}$ to the decoder network at each time step. This gives the decoder access to the context outside the to-be-normalized token at each time step but increases the dimensionality of the decoder input. An alternative way is to only use the attention-based vector as input to the decoder network ($c_i, t = c^{\text{mid}}_{i,t}$) and use the constant context only to initialize (`init`) the decoder RNN state. The difference between the `concat` and the `init` strategies is illustrated in Fig. 8.3. We will compare both methods in terms of speed and accuracy.

Architectures	Context strategy	Decoding speed	ALL			NON_TRIVIAL		
			English	Russian	Sinhala	English	Russian	Sinhala
`std`	-	4.33ms	2.19%	0.28%	0.07%	23.55%	1.91%	2.89%
`multilevel`	`concat`	2.40ms	2.23%	0.30%	0.10%	24.85%	2.49%	3.98%
`multilevel`	`init`	1.91ms	2.07%	0.29%	0.08%	22.66%	2.38%	3.07%
`ffcontext`	`concat`	2.44ms	2.10%	0.26%	0.08%	23.35%	2.10%	3.26%
`ffcontext`	`init`	2.09ms	2.06%	0.23%	0.08%	22.77%	1.86%	3.07%

Table 8.3 Error rates and decoding speeds of our neural text normalization models. ALL error rates are computed on all tokens, NON_TRIVIAL is restricted to non-trivial samples (i.e. no < `self` > token).

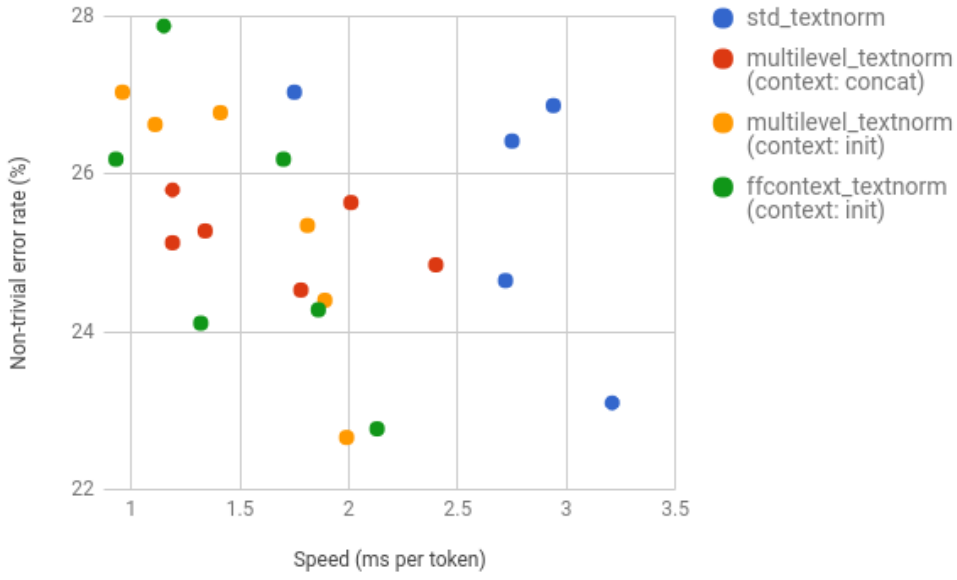


Fig. 8.4 Accuracy vs. speed for our different neural architectures in different sizes.

8.3.2 Unconstrained Neural Text Normalization

We first test our models under unconstrained beam search. We use 10M English and Russian training sentences as described by Sproat and Jaitly (2016). For English we constructed a test set of challenging text normalization problems by searching for specific phenomena in the held out set of the one billion words benchmark corpus (Chelba et al., 2014). We also report the performance of our models on a much smaller proprietary Sinhala data set. We refer to Zhang et al. (2019a) for details about model hyper-parameters and pre-processing.

Tab. 8.3 shows the error rates of the models on English, Russian and Sinhala. Decoding speeds are measured in milliseconds per token, i.e. all quantities are the lower the better. We report the median of the average (batch) CPU decoding speed (9 runs). The batch size is

Architectures	No context	1-Window	2-Window	3-Window	Unlimited
`std_textnorm`		24.17%	24.00%	23.55%	23.25%
`multilevel_textnorm`	28.19%	25.34%	25.69%	25.07%	24.85%
`ffcontext_textnorm`		24.54%	24.64%	22.77%	-

Table 8.4 Impact of the context window size on the NON_TRIVIAL error rates for English. We use the `concat` context strategy for `multilevel_textnorm` and the `init` strategy for `ffcontext_textnorm`.

optimized for each setup separately.[2] All models yield similar error rates across the board, but the decoding speeds differ significantly. The `init` strategy for using the context speeds up decoding since the computation required at each decoding time step is reduced. For example, our `multilevel_textnorm` model is 2.3 times faster than the baseline `std_textnorm` and improves the error rates on English slightly at the same time. Varying layer sizes in the neural network is a way to trade-off decoding speed and accuracy. Fig. 8.4 shows this trade-off for the different models. The `multilevel_textnorm` and `ffcontext_textnorm` models have better operation curves than `std_textnorm` (blue dots).

Context window size Our experiments in the previous section showed that the performance difference between the model with unlimited context width (`multilevel_textnorm`) and the windowed models (`std_textnorm` and `ffcontext_textnorm`) is rather small. This tends to confirm that modelling long-range context is often not crucial for text normalization. In order to study the impact of the context size on the performance we artificially limited the context width in the `multilevel_textnorm` to a window around the to-be-normalized token and compared it with the other models. The results in Tab. 8.4 suggest that even a window size of 1 can already yield reasonable performance on English, but using no context at all (first column) performs significantly worse.

Using Multiple Tokenization Levels One main advantage of the `multilevel_textnorm` model is that it allows to use different tokenizations for the context and the to-be-normalized (mid) token. We investigate three different tokenization strategies:

- Character: Character-level tokenization (character set size: 300)
- Word pieces: Subword-unit based tokenization. We chose a vocabulary size of 5,000 word pieces.

[2] Testing environment: Goobuntu 14.04.1, Linux 4.4.0 kernel, 12-core Intel Xeon CPU E5-1650 v4 at 3.60GHz, 16 RAM

Error rates (NON_TRIVIAL)

		T_{mid} Char	T_{mid} Subword	T_{mid} Word
T_{context}	Char	25.26%	27.78%	52.21%
	Subword	24.85%	28.06%	51.40%
	Word	25.28%	27.91%	52.81%

Decoding speed (milliseconds per token)

		T_{mid} Char	T_{mid} Subword	T_{mid} Word
T_{context}	Char	2.69ms	2.11ms	1.37ms
	Subword	2.40ms	1.72ms	1.21ms
	Word	2.34ms	1.74ms	1.22ms

Table 8.5 Using different tokenization strategies for $\text{tok}_{\text{context}}(\cdot)$ (rows) and $\text{tok}_{\text{mid}}(\cdot)$ (columns) in the `multilevel_textnorm` (`concat`) model for English.

- Words: Split at whitespace. We use a short list of the 30,000 most frequent words in the training corpus, and replace all other tokens with the special token UNK.

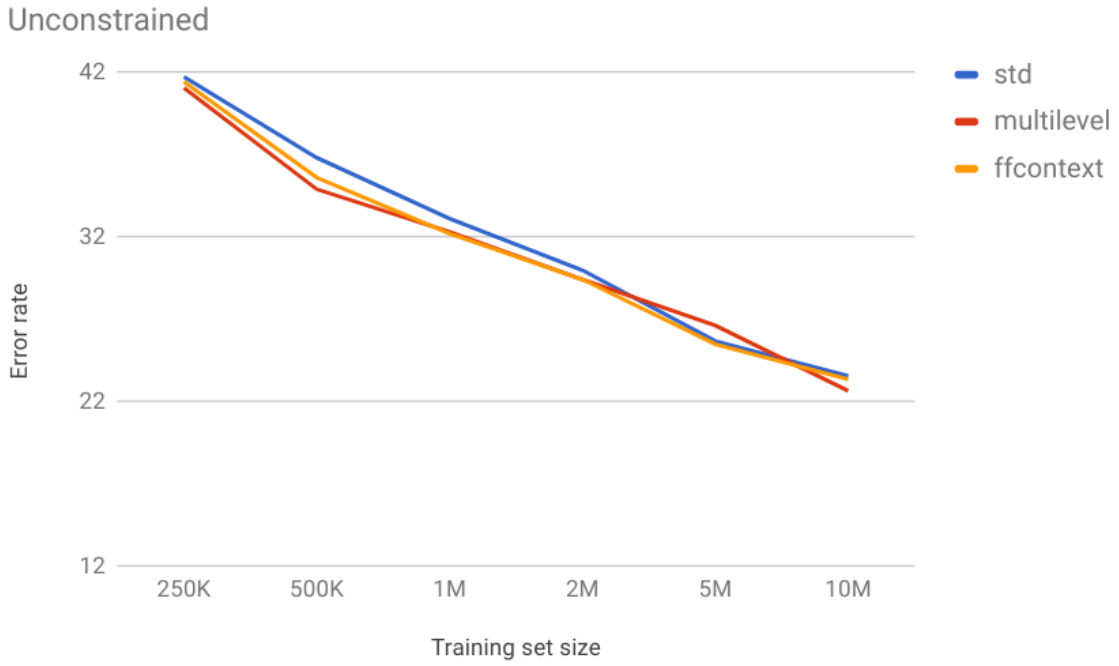
Tab. 8.5 shows the error rates on the test set. The row represents the context tokenization, and the column the tokenization for the to-be-normalized segment. Using characters for the mid segment works best (first column). However, the tokenization strategy for the context has only a minor impact on the error rate. The tokenization strategies have a clear impact on the decoding speed as coarse grained tokenization tends to be faster.

8.3.3 Constraining Neural Text Normalization with Covering Grammars

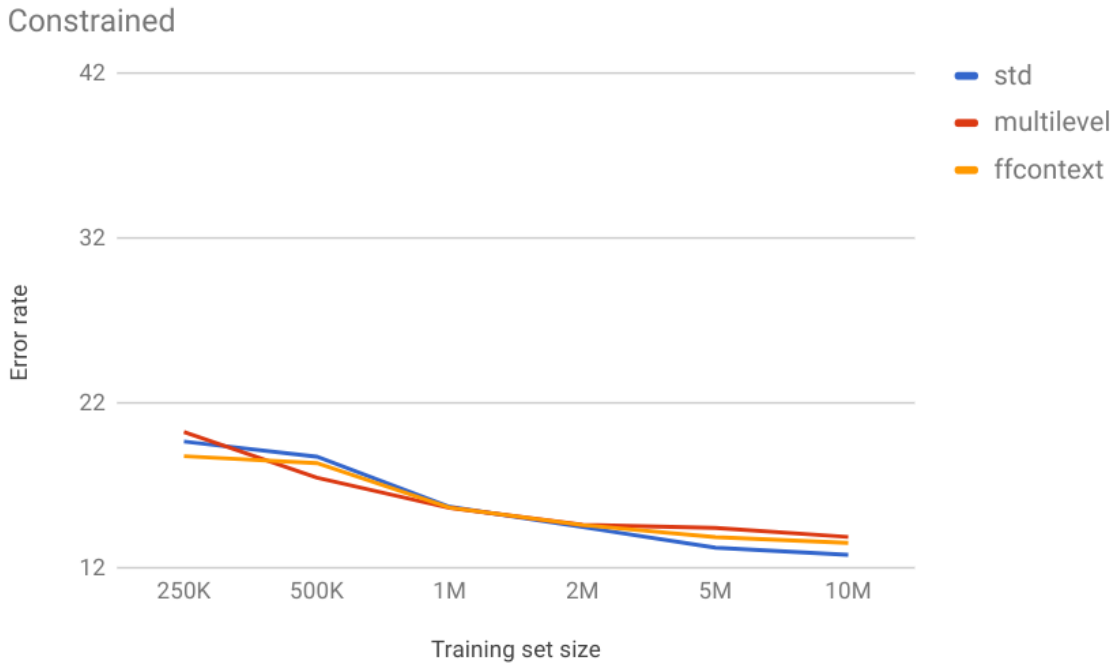
Despite very good overall error rates, all neural models we have discussed suffer from occasional 'catastrophic' errors – the sort of errors from which it is impossible to recover, such as misrepresenting the value of a number or outputting the wrong unit of measure. Sproat and Jaitly (2016) proposed to constrain the neural decoder with FSTs that are designed to prevent

	ALL		NON_TRIVIAL	
	Unconstrained	Constrained	Unconstrained	Constrained
`std_textnorm`	2.19%	**1.24%**	23.55%	**12.79%**
`multilevel_textnorm`	2.07%	**1.29%**	22.66%	**13.88%**
`ffcontext_textnorm`	2.06%	**1.26%**	22.77%	**13.75%**

Table 8.6 Constraining English neural text normalization models with covering grammars.



(a) Unconstrained NON_TRIVIAL error rates.



(b) Constrained NON_TRIVIAL error rates.

Fig. 8.5 The constrained systems are able to make better use of small training sets.

these kinds of errors. The approach is very similar to our hybrid SMT-NMT system from Sec. 4.3, with the difference that the FSTs are not generated by an SMT system but by parsing the input with a formal covering grammar. Covering grammars contain hand engineered rules which map input strings to the set of all possible vocalizations. Technically, the constraining FST is constructed on-the-fly by composing the input word x_i with a manually written Thrax grammar (Roark et al., 2012) and projecting on the output. Note that covering grammars cover the output rather than the input – i.e. they try to make sure that *if* the input string can be parsed, the correct output is covered by the grammar. They do not guarantee that every input string is parsable. For example, 32.5% of the non-trivial English training examples in our English experiments cannot be parsed by the grammar. We back off to unconstrained decoding for all input strings which are not accepted. Constraining the neural decoder with the space spanned by the covering grammar guarantees to prevent catastrophic errors for all input strings accepted by the grammar.

All experiments are performed with English. Tab. 8.6 shows the gains from constraining the neural output to the covering grammar. Constraining the output consistently yields large gains over the pure neural systems.

Training data size A major challenge in text normalization is the lack of annotated training data. Therefore, model performance in low resource settings is an important evaluation criterion. Fig. 8.5 plots the NON_TRIVIAL error rates of the three neural models over the training data size for both unconstrained and constrained decoding. The covering grammar is able to correct a large amount of errors made by the neural models, particularly when the training set is small. Even the grammar-based approach with only 250K training tokens cannot be outperformed by a purely neural model trained on 10M tokens.

Correlation between unconstrained and constrained error rates The scatter chart in Fig. 8.6 shows the correlation of the NON_TRIVIAL error rates between unconstrained and constrained decoding for several models of varying sizes. Surprisingly, this correlation seems to be rather weak – unconstrained model performance is not a good indicator for the performance under constraints. For some models, the grammar tends to compensate for differences in unconstrained decoding. For example, the dynamic range of the unconstrained error rate for `ffcontext_textnorm` is $[23.37\%, 25.91\%]$, but the constrained error rates are all within $[13.66\%, 14.46\%]$.

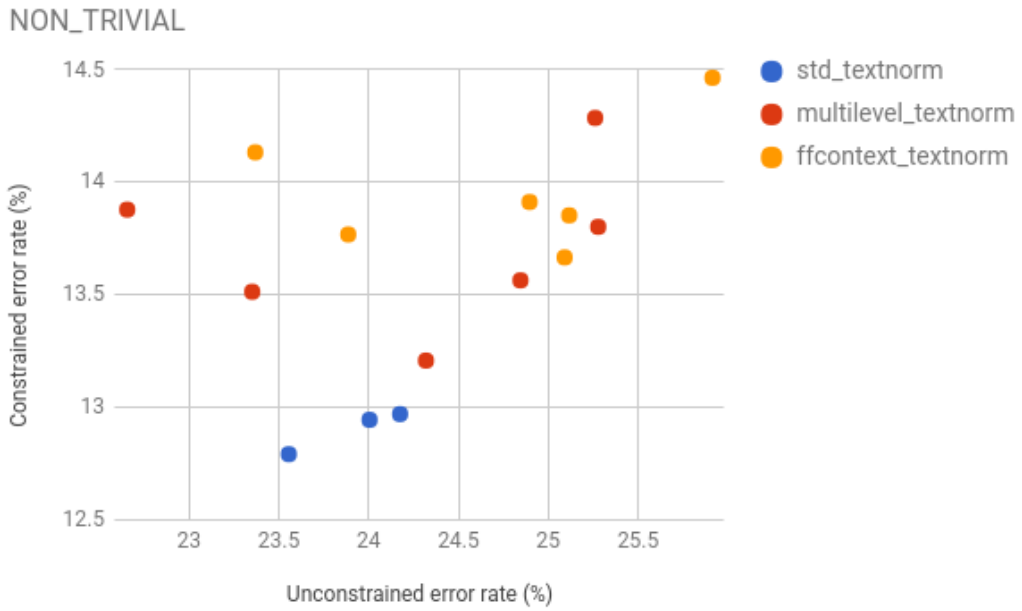


Fig. 8.6 A scatter plot of constrained and unconstrained NON_TRIVIAL error rates for our three architectures in different sizes. Constrained and unconstrained error rates are only weakly correlated.

8.4 Operation Sequence Neural Machine Translation

Neural machine translation (NMT) models (Ch. 3) are remarkably effective in modelling the distribution over target sentences conditioned on the source sentence, and yield superior translation performance compared to traditional statistical machine translation (SMT) on many language pairs. However, as we argued in Sec. 3.12, it is often difficult to extract a comprehensible explanation for the predictions of these models as information in the network is represented by real-valued vectors or matrices (Ding et al., 2017). In contrast, the translation process in SMT is 'transparent' as it can identify the source word which caused a target word through word alignment. Most NMT models do not use the concept of word alignment. It is tempting to interpret encoder-decoder attention matrices in neural models (Sec. 3.6.1) as (soft) alignments, but previous work has found that the attention weights in NMT are often erratic and differ significantly from traditional word alignments:

- "The attention model for NMT does not always fulfill the role of a word alignment model, but may in fact dramatically diverge." (Koehn and Knowles, 2017)

- "We perform extensive experiments across a variety of NLP tasks that aim to assess the degree to which attention weights provide meaningful 'explanations' for predictions. We find that they largely do not." (Jain and Wallace, 2019)
- "Attention weights are only noisy predictors of even intermediate components' importance, and should not be treated as justification for a decision." (Serrano and Smith, 2019)
- "Although attention is very useful for under-standing the connection between source and target words, only using attention is not sufficient for deep interpretation of target word generation." (Ding et al., 2017)
- "Attention agrees with traditional alignments to a high degree in the case of nouns. However, it captures other information rather than only the translational equivalent in the case of verbs." (Ghader and Monz, 2017)
- "Attention visualizations are misleading and should be treated with care when explaining the underlying deep learning system." (Brunner et al., 2019)

We will discuss the difference between attention and alignment further in Sec. 8.4.6. Our goal in this section is explainable NMT by developing a transparent translation process for neural models. Our approach does not change the neural architecture, but represents the translation together with its alignment as a linear sequence of operations. The neural model predicts this operation sequence, and thus simultaneously generates a translation and an explanation for it in terms of alignments from the target words to the source words that generate them. The operation sequence is "self-explanatory"; it does not explain an underlying NMT system but is rather a single representation produced by the NMT system that can be used to generate translations along with an accompanying explanatory alignment to the source sentence. We report competitive results of our method on Spanish-English, Portuguese-English, and Japanese-English, with the benefit of producing hard alignments for better interpretability. We discuss the theoretical connection between our approach and hierarchical SMT (Sec. 2.7) by showing that an operation sequence can be seen as a derivation in a formal grammar.

8.4.1 A Neural Operation Sequence Model

Our operation sequence neural machine translation (OSNMT) model is inspired by the operation sequence model for SMT (Durrani et al., 2011), but changes the set of operations to be more appropriate for neural sequence models. OSNMT is not restricted to a particular architecture, i.e. any seq2seq model such as RNN-based, convolutional, or self-attention-based models

(Sec. 3.6.6) could be used. In this work, we use the Transformer architecture (Vaswani et al., 2017) in all experiments.

In OSNMT, the neural seq2seq model learns to produce a sequence of operations. An OSNMT operation sequence describes a translation (the 'compiled' target sentence) and explains each target token with a hard link into the source sentence. OSNMT keeps track of the positions of a source-side read head and a target-side write head. The read head monotonically walks through the source sentence, whereas the position of the write head can be moved from marker to marker in the target sentence. OSNMT defines the following operations to control head positions and produce output words.

- `POP_SRC`: Move the read head right by one token.
- `SET_MARKER`: Insert a marker symbol into the target sentence at the position of the write head.
- `JMP_FWD`: Move the write head to the nearest marker right of the current head position in the target sentence.
- `JMP_BWD`: Move the write head to the nearest marker left of the current head position in the target sentence.
- `INSERT`(t): Insert a target token t into the target sentence at the position of the write head.

Tab. 8.7 illustrates the generation of a Japanese-English translation in detail. The neural seq2seq model is trained to produce the sequence of operations in the first column of Tab. 8.7. The initial state of the target sentence is a single marker symbol X_1. Generative operations like `SET_MARKER` or `INSERT`(t) insert a single symbol left of the current marker (highlighted). The model begins with a `SET_MARKER` operation, which indicates that the translation of the first word in the source sentence is not at the beginning of the target sentence. Indeed, after "translating" the identities '2000' and 'hr', in time step 6 the model jumps back to the marker X_2 and continues writing left of '2000'. The translation process terminates when the read head is at the end of the source sentence. The final translation in plain text can be obtained by removing all markers from the (compiled) target sentence.

8.4.2 OSNMT Represents Alignments

The word alignment can be derived from the operation sequence by looking up the position of the read head for each generated target token. The alignment for the example in Tab. 8.7 is shown in Fig. 8.7. Note that similarly to the IBM models (Sec. 2.3) and the OSM for

	Operation	Source sentence	Target sentence (compiled)
		2000 hr の 安定 動作 を 確認 し た	X_1
1	SET_MARKER	2000 hr の 安定 動作 を 確認 し た	X_2 X_1
2	2000	2000 hr の 安定 動作 を 確認 し た	X_2 2000 X_1
3	POP_SRC	2000 hr の 安定 動作 を 確認 し た	X_2 2000 X_1
4	hr	2000 hr の 安定 動作 を 確認 し た	X_2 2000 hr X_1
5	POP_SRC	2000 hr の 安定 動作 を 確認 し た	X_2 2000 hr X_1
6	JMP_BWD	2000 hr の 安定 動作 を 確認 し た	X_2 2000 hr X_1
7	SET_MARKER	2000 hr の 安定 動作 を 確認 し た	X_3 X_2 2000 hr X_1
8	of	2000 hr の 安定 動作 を 確認 し た	X_3 of X_2 2000 hr X_1
9	POP_SRC	2000 hr の 安定 動作 を 確認 し た	X_3 of X_2 2000 hr X_1
10	JMP_BWD	2000 hr の 安定 動作 を 確認 し た	X_3 of X_2 2000 hr X_1
11	stable	2000 hr の 安定 動作 を 確認 し た	stable X_3 of X_2 2000 hr X_1
12	POP_SRC	2000 hr の 安定 動作 を 確認 し た	stable X_3 of X_2 2000 hr X_1
13	operation	2000 hr の 安定 動作 を 確認 し た	stable operation X_3 of X_2 2000 hr X_1
14	POP_SRC	2000 hr の 安定 動作 を 確認 し た	stable operation X_3 of X_2 2000 hr X_1
15	JMP_FWD	2000 hr の 安定 動作 を 確認 し た	stable operation X_3 of X_2 2000 hr X_1
16	JMP_FWD	2000 hr の 安定 動作 を 確認 し た	stable operation X_3 of X_2 2000 hr X_1
17	was	2000 hr の 安定 動作 を 確認 し た	stable operation X_3 of X_2 2000 hr was X_1
18	POP_SRC	2000 hr の 安定 動作 を 確認 し た	stable operation X_3 of X_2 2000 hr was X_1
19	POP_SRC	2000 hr の 安定 動作 を 確認 し た	stable operation X_3 of X_2 2000 hr was X_1
20	confirmed	2000 hr の 安定 動作 を 確認 し た	stable operation X_3 of X_2 2000 hr was confirmed X_1
21	POP_SRC	2000 hr の 安定 動作 を 確認 し た	stable operation X_3 of X_2 2000 hr was confirmed X_1

Table 8.7 Generation of the target sentence "stable operation of 2000 hr was confirmed". The neural model produces the linear sequence of operations in the first column. The positions of the source-side read head and the target-side write head are highlighted. The marker in the target sentence produced by the i-th SET_MARKER operation is denoted with 'X_{i+1}'; X_1 is the initial marker. We denote INSERT(t) operations as t to simplify notation.

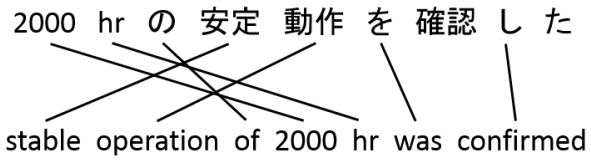


Fig. 8.7 The translation and the alignment derived from the operation sequence in Tab. 8.7.

SMT (Durrani et al., 2011), our OSNMT can only represent 1:n alignments. Thus, each target token is aligned to exactly one source token, but a source token can generate any number of (possibly non-consecutive) target tokens.

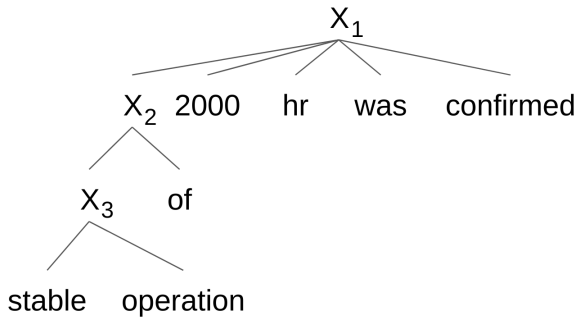


Fig. 8.8 Target-side tree representation of the operation sequence in Tab. 8.7.

8.4.3 OSNMT Represents Hierarchical Structure

We can also derive a tree structure from the operation sequence in Tab. 8.7 (Fig. 8.8) in which each marker is represented by a nonterminal node with outgoing arcs to symbols inserted at that marker. The target sentence can be read off the tree by depth-first search traversal (post-order).

More formally, synchronous context-free grammars (SCFGs) generate pairs of strings by pairing two context-free grammars. Phrase-based hierarchical SMT (Chiang, 2007) uses SCFGs to model the relation between the source sentence and the target sentence. Multitext grammars (MTGs) are a generalization of SCFGs to more than two output streams (Melamed, 2003; Melamed et al., 2004). We find that an OSNMT sequence can be interpreted as a sequence of rules of a tertiary MTG $\mathscr{G}$ which generates 1.) the source sentence, 2.) the target sentence, and 3.) the position of the target side write head. The start symbol of $\mathscr{G}$ is

$$[(S),(X_1),(P_1)]^{\mathsf{T}} \tag{8.5}$$

which initializes the source sentence stream with a single nonterminal S, the target sentence with the initial marker X_1 and the position of the write head with 1 (P_1). Following Melamed et al. (2004) we denote rules in $\mathscr{G}$ as

$$[(\alpha_1),(\alpha_2),(\alpha_3)]^{\mathsf{T}} \rightarrow [(\beta_1),(\beta_2),(\beta_3)]^{\mathsf{T}} \tag{8.6}$$

where $\alpha_1,\alpha_2,\alpha_3$ are single nonterminals or empty, β_1,β_2,β_3 are strings of terminals and nonterminals, and $\alpha_k \rightarrow \beta_k$ for all $k \in \{1,2,3\}$ with nonempty α_i are the rewriting rules for each of the three individual components which need to be applied simultaneously. `POP_SRC` extends the source sentence prefix in the first stream by one token.

$$\texttt{POP\_SRC} : \forall s \in \mathscr{V}_{src} : \begin{bmatrix}(S)\\()\\()\end{bmatrix} \rightarrow \begin{bmatrix}(sS)\\()\\()\end{bmatrix} \tag{8.7}$$

Derivation	OSNMT
$[(S),(X_1),P_1]^\mathsf{T}$	
$\overset{\text{Eq. 8.7}}{\rightarrow} [(2000\ S),(X_1),P_1]^\mathsf{T}$	SET_MARKER
$\overset{\text{Eq. 8.9}}{\rightarrow} [(2000\ S),(X_2X_1),(P_1)]^\mathsf{T}$	2000
$\overset{\text{Eq. 8.10}}{\rightarrow} [(2000\ S),(X_2\ 2000\ X_1),(P_1)]^\mathsf{T}$	POP_SRC
$\overset{\text{Eq. 8.7}}{\rightarrow} \begin{bmatrix}(2000\ \text{hr}\ S)\\(X_2\ 2000\ X_1)\\(P_1)\end{bmatrix}$	hr
$\overset{\text{Eq. 8.10}}{\rightarrow} \begin{bmatrix}(2000\ \text{hr}\ S)\\(X_2\ 2000\ \text{hr}\ X_1)\\(P_1)\end{bmatrix}$	POP_SRC
$\overset{\text{Eq. 8.7}}{\rightarrow} \begin{bmatrix}(2000\ \text{hr}\ \curvearrowleft\ S)\\(X_2\ 2000\ \text{hr}\ X_1)\\(P_1)\end{bmatrix}$	JMP_BWD
$\overset{\text{Eq. 8.8}}{\rightarrow} \begin{bmatrix}(2000\ \text{hr}\ \curvearrowleft\ S)\\(X_2\ 2000\ \text{hr}\ X_1)\\(1\ P_2)\end{bmatrix}$	SET_MARKER
$\overset{\text{Eq. 8.9}}{\rightarrow} \begin{bmatrix}(2000\ \text{hr}\ \curvearrowleft\ S)\\(X_3X_2\ 2000\ \text{hr}\ X_1)\\(1\ P_2)\end{bmatrix}$	of
$\overset{\text{Eq. 8.10}}{\rightarrow} \begin{bmatrix}(2000\ \text{hr}\ \curvearrowleft\ S)\\(X_3\ \text{of}\ X_2\ 2000\ \text{hr}\ X_1)\\(1\ P_2)\end{bmatrix}$	...
...	

Table 8.8 Derivation in $\mathscr{G}$ for the example of Tab. 8.7.

where $\mathscr{V}_{src}$ is the source language vocabulary. A jump from marker X_i to X_j is realized by replacing P_i with P_j in the third grammar component:

$$\texttt{JMP} : \forall i,j \in \mathscr{N} : [(),(),P_i]^\mathsf{T} \rightarrow [(),(),(iP_j)]^\mathsf{T} \tag{8.8}$$

where $\mathscr{N} = \mathbb{G}_n$ is the set of the first n natural numbers for a sufficiently large n ($\mathbb{G}_n$ is defined in Sec. 1.4). The generative operations (SET_MARKER and INSERT(t)) insert symbols into the second component.

$$\texttt{SET\_MARKER} : \forall i \in \mathcal{N} : \begin{bmatrix} () \\ (X_i) \\ (P_i) \end{bmatrix} \rightarrow \begin{bmatrix} () \\ (X_{i+1}X_i) \\ (P_i) \end{bmatrix} \tag{8.9}$$

$$\texttt{INSERT} : \forall i \in \mathcal{N}, t \in \Sigma_{trg} : \begin{bmatrix} () \\ (X_i) \\ (P_i) \end{bmatrix} \rightarrow \begin{bmatrix} () \\ (tX_i) \\ (P_i) \end{bmatrix} \tag{8.10}$$

where Σ_{trg} is the target language vocabulary. The identity mapping $P_i \rightarrow P_i$ in the third component enforces that the write head is at marker X_i. We note that $\mathscr{G}$ is not only context-free but also regular in the first and third components (but not in the second component due to Eq. 8.9). Rules of the form in Eq. 8.10 are directly related to alignment links (cf. Fig. 8.7) as they represent the fact that target token t is aligned to the last terminal symbol in the first stream. We formalize removing markers/nonterminals at the end by introducing a special nonterminal T which is eventually mapped to the end-of-sentence symbol EOS:

$$[(S),(),()]^{\mathsf{T}} \rightarrow [(T),(),()]^{\mathsf{T}} \tag{8.11}$$

$$[(T),(),()]^{\mathsf{T}} \rightarrow [(\text{EOS}),(),()]^{\mathsf{T}} \tag{8.12}$$

$$\forall i \in \mathcal{N} : [(T),(X_i),()]^{\mathsf{T}} \rightarrow [(T),(\varepsilon),()]^{\mathsf{T}} \tag{8.13}$$

$$\forall i \in \mathcal{N} : [(T),(),(P_i)]^{\mathsf{T}} \rightarrow [(T),(),(\varepsilon)]^{\mathsf{T}} \tag{8.14}$$

Tab. 8.8 illustrates that there is a 1:1 correspondence between a derivation in $\mathscr{G}$ and an OSNMT operation sequence. The target-side derivation (the second component in $\mathscr{G}$) is structurally similar to a binarized version of the tree in Fig. 8.8. However, we assign scores to the structure via the corresponding OSNMT sequence which does not need to obey the usual conditional independence assumptions in hierarchical SMT. Therefore, even though $\mathscr{G}$ is context-free in the second component, our scoring model for $\mathscr{G}$ is more powerful as it conditions on the OSNMT history which potentially contains context information. Note that OSNMT is deficient (Brown et al., 1993) as it assigns non-zero probability mass to any operation sequence, not only those with derivation in G.

We further note that subword-based OSNMT can potentially represent any alignment to any target sentence as long as the alignment does not violate the 1:n restriction. This is in contrast to phrase-based SMT where reference translations often do not have a derivation in the SMT system due to coverage problems (Auli et al., 2009).

8.4.4 Comparison to the OSM for SMT

Our OSNMT set of operations (`POP_SRC`, `SET_MARKER`, `JMP_FWD`, `JMP_BWD`, and `INSERT`(t)) is inspired by the original OSM for SMT (Durrani et al., 2011) as it also represents the translation process as linear sequence of operations. However, there are significant differences which make OSNMT more suitable for neural models. First, OSNMT is monotone on the source side, and allows jumps on the target side. SMT-OSM operations jump in the source sentence. We argue that source side monotonicity potentially mitigates coverage issues of neural models (Sec. 3.10.1) as the attention can learn to scan the source sentence from left to right. Another major difference is that we use *markers* rather than *gaps*, and do not close a gap/marker after jumping to it. This is an implication of OSNMT jumps being defined on the target side since the size of a span is unknown at inference time.

Algorithm 13 Align2OSNMT(a, **x**, **y**)

$holes \leftarrow \{(0,\infty)\}$
$ops \leftarrow \langle\rangle$ {Initialize with empty list}
$head \leftarrow 0$
for $i \leftarrow 1$ **to** $|\mathbf{x}|$ **do**
  for all $j \in \{j|a_j = i\}$ **do**
    $hole\_idx \leftarrow holes$.find($j$)
    $d \leftarrow hole\_idx - head$
    if $d < 0$ **then**
      ops.extend(`JMP_BWD`.repeat($-d$))
    end if
    if $d > 0$ **then**
      ops.extend(`JMP_FWD`.repeat(d))
    end if
    $head \leftarrow hole\_idx$
    $(s,t) \leftarrow holes[head]$
    if $s \neq j$ **then**
      $holes$.append($(s, j-1)$)
      $head \leftarrow head + 1$
      ops.append(`SET_MARKER`)
    end if
    ops.append($\mathtt{y_j}$)
    $holes[head] \leftarrow (j+1,t)$
  end for
  ops.append(`SRC_POP`)
end for
return ops

Corpus	Language pair	# Sentences
Scielo	Spanish-English	587K
Scielo	Portuguese-English	513K
WAT	Japanese-English	1M

Table 8.9 Training set sizes.

8.4.5 Training

We train our Transformer model as usual by minimizing the negative log-likelihood of the target sequence. However, in contrast to plain text NMT, the target sequence is not a plain sequence of subword or word tokens but a sequence of operations. Consequently, we need to map the target sentences in the training corpus to OSNMT representations. We first run a statistical word aligner like Giza++ (Och and Ney, 2003) to obtain an aligned training corpus. We delete all alignment links which violate the 1:n restriction of OSNMT (cf. Sec. 8.4.1). The alignments together with the target sentences are then used to generate the reference operation sequences for training. The algorithm for this conversion is shown in Alg. 13.[3]

Note that an operation sequence represents one specific alignment, which means that the only way for an OSNMT sequence to be generated correctly is if both the word alignment and the target sentence are also correct. Thereby, the neural model learns to align and translate at the same time. However, there is spurious ambiguity as one alignment can be represented by different OSNMT sequences. For instance, simply adding a `SET_MARKER` operation at the end of an OSNMT sequence does not change the alignment represented by it.

8.4.6 Results

We evaluate on three language pairs: Japanese-English (ja-en), Spanish-English (es-en), and Portuguese-English (pt-en). We use the ASPEC corpus (Nakazawa et al., 2016) for ja-en and the health science portion of the Scielo corpus (Neves et al., 2016) for es-en and pt-en. Training set sizes are summarized in Tab. 8.9. We use byte pair encoding (Sennrich et al., 2016c) with 32K merge operations for all systems (joint encoding models for es-en and pt-en and separate source/target models for ja-en). We trained Transformer models (Vaswani et al., 2017)[4] until convergence (250K steps for plain text, 350K steps for OSNMT) on a single GPU using Tensor2Tensor (Vaswani et al., 2018) after removing sentences with more than

[3] A Python implementation is available at https://github.com/fstahlberg/ucam-scripts/blob/master/t2t/align2osm.py.

[4] We follow the `transformer_base` configuration and use 6 layers, 512 hidden units, and 8 attention heads in both the encoder and decoder.

Method	BLEU	
	es-en	pt-en
Align on subword level	36.7	38.1
Convert word level alignments	37.1	38.4

Table 8.10 Generating training alignments on the subword level.

250 tokens. Batches contain around 4K source and 4K target tokens. Transformer training is very sensitive to the batch size and the number of GPUs (Popel and Bojar, 2018). Therefore, we delay SGD updates (Saunders et al., 2018) to every 8 steps to simulate 8 GPU training as recommended by Vaswani et al. (2017). Based on the performance on the ja-en dev set we decode the plain text systems with a beam size of 4 and OSNMT with a beam size of 8 using our SGNMT decoder (Ch. 5). We use length normalization for ja-en but not for es-en or pt-en. We report cased `multi-bleu.pl` BLEU scores on the tokenized text to be comparable with the WAT evaluation campaign on ja-en.[5].

Generating training alignments As outlined in Sec. 8.4.5 we use Giza++ (Och and Ney, 2003) to generate alignments for training OSNMT. We experimented with two different methods to obtain alignments on the subword level. First, Giza++ can directly align the source-side subword sequences to target-side subword sequences. Alternatively, we can run Giza++ on the word level, and convert the word alignments to subword alignments in a postprocessing step by linking subwords if the words they belong to are aligned with each other. Tab. 8.10 compares both methods and shows that converting word alignments is marginally better. Thus, we use this method in all other experiments.

Constrained beam search Unconstrained neural decoding can yield invalid OSNMT sequences. For example, the `JMP_FWD` and `JMP_BWD` operations are undefined if the write head is currently at the position of the last or first marker, respectively. The number of `SRC_POP` operations must be equal to the number of source tokens in order for the read head to scan the entire source sentence. Therefore, we constrain these operations during decoding. We have implemented the constraints in our publicly available SGNMT decoding platform (Ch. 5). However, these constraints are only needed for a small fraction of the sentences. Tab. 8.11 shows that even unconstrained decoding yields valid OSNMT sequences in 92.49% of the cases.

[5] http://lotus.kuee.kyoto-u.ac.jp/WAT/evaluation/list.php?t=2&o=4

Type	Frequency
Valid	**92.49%**
Not enough SRC_POP	7.28%
Too many SRC_POP	0.22%
Write head out of range	0.06%

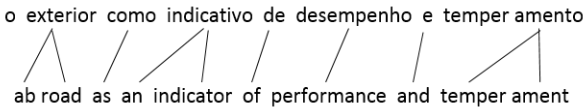
Table 8.11 Frequency of invalid OSNMT sequences produced by an unconstrained decoder on the ja-en test set.

Representation	BLEU			
	es-en	pt-en	ja-en	
			dev	test
Plain	37.6	37.5	28.3	28.1
OSNMT	37.1	38.4	28.1	28.8

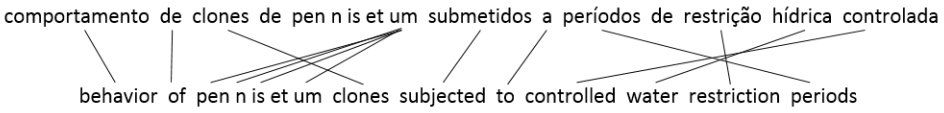
Table 8.12 Comparison between plain text and OSNMT on Spanish-English (es-en), Portuguese-English (pt-en), and Japanese-English (ja-en).

Comparison with plain text NMT Tab. 8.12 compares our OSNMT systems with standard plain text models on all three language pairs. OSNMT performs better on the pt-en and ja-en test sets, but slightly worse on es-en. We think that more engineering work such as optimizing the set of operations or improving the training alignments could lead to more consistent gains from using OSNMT. However, we leave this to future work since the main motivation for our method is explainable NMT and not primarily improving translation quality.

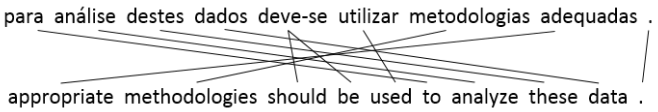
Alignment quality Tab. 8.13 contains example translations and subword-alignments generated from our Portuguese-English OSNMT model. Alignment links from source words consisting of multiple subwords are mapped to the final subword, visible for the words 'temperamento' in the first example and 'pennisetum' in the second one. The length of the operation sequences increases with alignment complexity as operation sequences for monotone alignments consist only of INSERT(t) and SRC_POP operations (example 1). However, even complex mappings are captured very well by OSNMT as demonstrated by the third example. Note that OSNMT can represent long-range reorderings very efficiently: the movement from 'para' in the first position to 'to' in the tenth position is simply achieved by starting the operation sequence with 'SET_MARKER to' and a JMP_BWD operation later. The first example in particular demonstrates the usefulness of such alignments as the wrong lexical choice ('abroad' rather than 'body shape') can be traced back to the source word 'exterior'.



Operation sequence: SRC_POP ab road_ SRC_POP as_ SRC_POP an_ indicator_ SRC_POP of_ SRC_POP performance_ SRC_POP and_ SRC_POP SRC_POP temper ament_ SRC_POP
Reference: the body shape as an indicative of performance and temperament



Operation sequence: behavior_ SRC_POP of_ SRC_POP SET_MARKER clones_ SRC_POP SRC_POP SRC_POP SRC_POP SRC_POP JMP_BWD pen n is et um_ SRC_POP JMP_FWD subjected_ SRC_POP to_ SRC_POP SET_MARKER periods_ SRC_POP SRC_POP JMP_BWD SET_MARKER restriction_ SRC_POP JMP_BWD SET_MARKER water_ SRC_POP JMP_BWD controlled_ SRC_POP
Reference: response of pennisetum clons to periods of controlled hidric restriction



Operation sequence: SET_MARKER to_ SRC_POP analyze_ SRC_POP these_ SRC_POP data_ SRC_POP JMP_BWD SET_MARKER should_ be_ SRC_POP used_ SRC_POP JMP_BWD SET_MARKER methodologies_ SRC_POP> JMP_BWD appropriate_ SRC_POP JMP_FWD JMP_FWD JMP_FWD ._ SRC_POP
Reference: to analyze these data suitable methods should be used .

Table 8.13 Examples of Portuguese-English translations together with their (subword-) alignments induced by the operation sequence. Alignment links from source words consisting of multiple subwords were mapped to the final subword in the training data, visible for 'temperamento' and 'pennisetum'.

For a qualitative assessment of the alignments produced by OSNMT we ran Giza++ to align the generated translations to the source sentences, enforced the 1:*n* restriction of OSNMT, and used the resulting alignments as reference for computing the alignment error rate (Och and Ney, 2003, AER). Fig. 8.9 shows that as training proceeds, OSNMT learns to both produce high quality translations (increasing BLEU score) and accurate alignments (decreasing AER).

As mentioned in the introduction, a light-weight way to extract 1:*n* alignments from a vanilla attentional LSTM-based seq2seq model is to take the maximum over attention weights for each

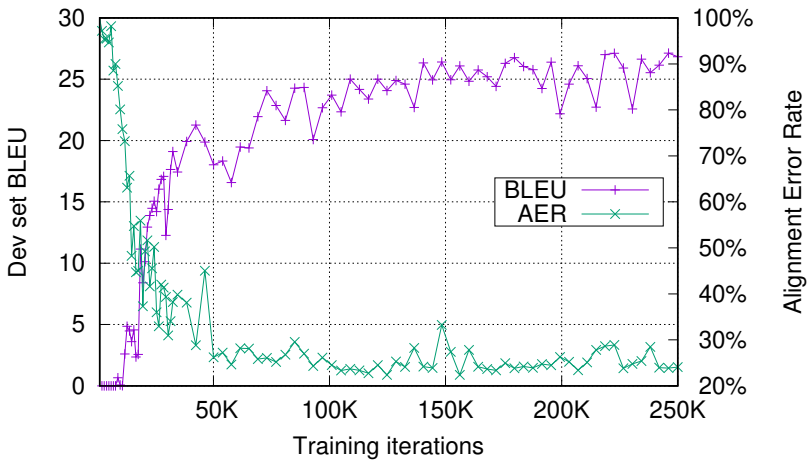


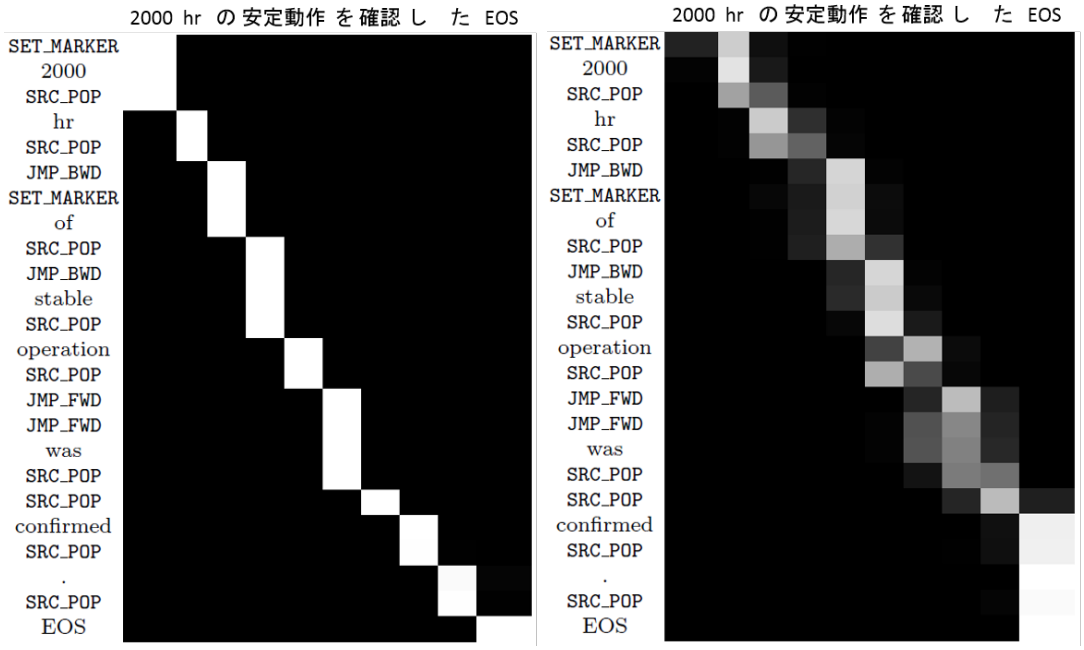
Fig. 8.9 AER and BLEU training curves for OSNMT on the Japanese-English dev set.

Representation	Alignment extraction	AER (in %) dev	test
Plain	LSTM forced decoding	63.9	63.7
Plain	LSTM forced decoding with supervised attention (Liu et al., 2016b, *Cross Entropy* loss)	54.9	54.7
OSNMT	OSNMT	24.2	21.5

Table 8.14 Comparison between OSNMT and using the attention matrix from forced decoding with a recurrent model.

target token. This is possible because, unlike the Transformer, LSTM-based models usually only have a single soft attention matrix. However, in our experiments, LSTM-based NMT was more than 4.5 BLEU points worse than the Transformer on Japanese-English. Therefore, to compare AERs under comparable BLEU scores, we used the LSTM-based models in forced decoding mode on the output of our plain text Transformer model from Tab. 8.12. We trained two different LSTM models: one standard model by optimizing the likelihood of the training set, and a second one with supervised attention following Liu et al. (2016b). Tab. 8.14 shows that the supervised attention loss of Liu et al. (2016b) improves the AER of the LSTM model. However, OSNMT is able to produce much better alignments since it generates the alignment along with the translation in a single decoding run.

OSNMT sequences contain target words in source sentence order An OSNMT sequence can be seen as a sequence of target words in source sentence order, interspersed with instructions on how to put them together to form a fluent target sentence. For example, if we strip out



(a) Layer 4, head 1; attending to the source side read head.

(b) Layer 2, head 3; attending to the right trigram context of the read head.

Fig. 8.10 Encoder-decoder attention weights.

all SRC_POP, SET_MARKER, JMP_FWD, and JMP_BWD operations in the OSNMT sequence in the second example of Tab. 8.13 we get:

> behavior of clones pennisetum subjected to periods restriction water controlled

The word-by-word translation back to Portugese is:

> comportamento de clones pennisetum submetidos a períodos restrição hídrica controlada

This restores the original source sentence (cf. Tab. 8.13) up to unaligned source words. Therefore, we can view the operations for controlling the write head (SET_MARKER, JMP_FWD, and JMP_BWD) as reordering instructions for the target words which appear in source sentence word order within the OSNMT sequence.

The role of multi-head attention In this paper, we use a standard seq2seq model (the Transformer architecture (Vaswani et al., 2017)) to generate OSNMT sequences from the source sentence. This means that our neural model is representation-agnostic: we do not explicitly incorporate the notion of read and write heads into the neural architecture. In

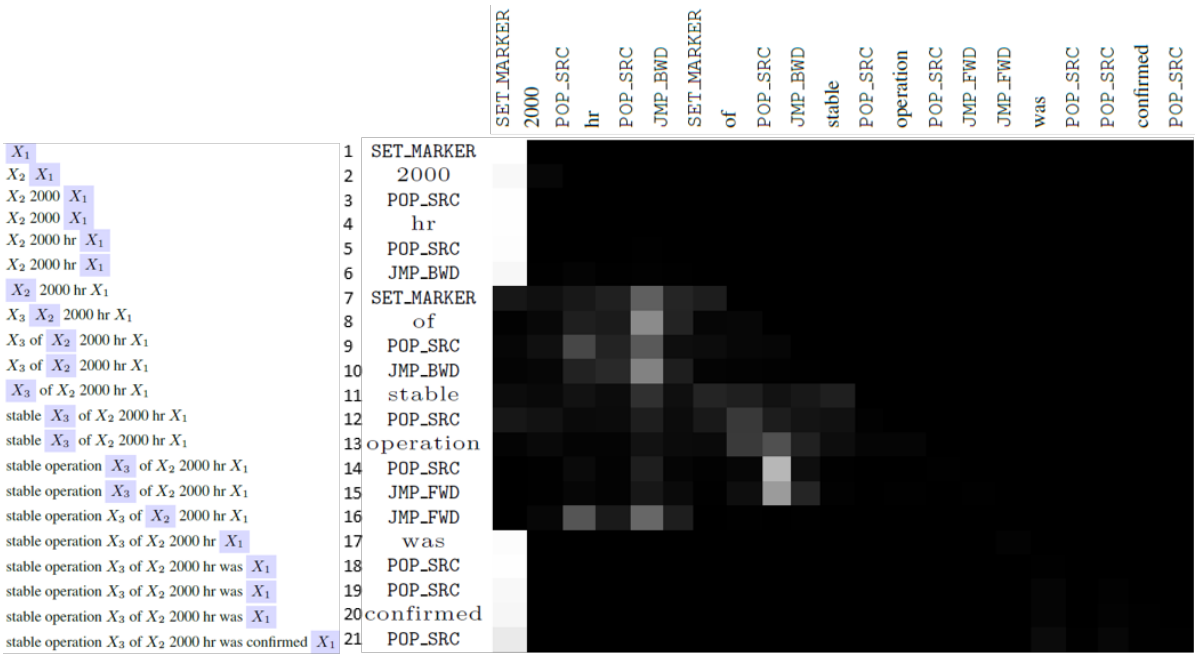


Fig. 8.11 Decoder self-attention weight matrix in layer 5, head 3; attending to the first position (constant) if the target-side write head (marked with blue) is at the end of the sentence (lines 1-5 and 17-21).

particular, neither in training nor in decoding do we explicitly bias the Transformer's attention layers towards consistency with the alignment represented by the OSNMT sequence. Our Transformer model has 48 encoder-decoder attention matrices due to multi-head attention (8 heads in each of the 6 layers). We have found that many of these attention matrices have strong and interpretable links to the translation process represented by the OSNMT sequence. For example, Fig. 8.10a shows that the first head in layer 4 follows the source-side read head position very closely: at each SRC_POP operation the attention shifts by one to the next source token. Other attention heads have learned to take other responsibilities. For instance, head 3 in layer 2 (Fig. 8.10b) attends to the trigram right of the source head.

The Transformer uses self-attention rather than recurrence in the decoder network to condition on the translation history. We observe that similarly to encoder-decoder attention heads relating to the OSNMT source-side read head position, some decoder self-attention heads emit interpretable behavior related to the target-side write head position. For example, head 3 in layer 5 (Fig. 8.11) attends to the first token in time steps 1-5 and 17-21, i.e. the initial decoder state which is constant. Fig. 8.11 illustrates that the write head (highlighted in blue) is at the end of the compiled target sentence in these time steps. Therefore, this attention layer encodes whether or not the OSNMT write head is at the end of the sentence. If the write head is in the middle of the sentence, the attention tends to be concentrated on tokens right of it.

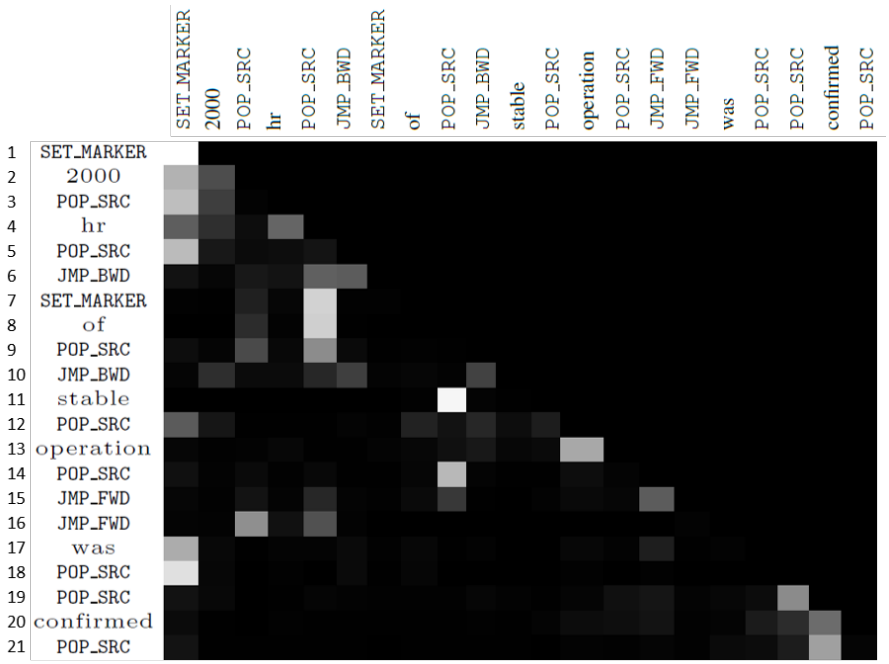


Fig. 8.12 Decoder self-attention weight matrix in layer 5, head 6 with long-range dependencies.

Fig. 8.12 shows that some attention matrices are able to capture very long range dependencies (white spots far away from the main diagonal). Note that in time step 16 the OSNMT write head jumps back to the end of the target sentence. We see a long range dependency from time step 16 to step 3 and 5 around which the words just before the new write head position ('2000' and 'hr') were produced.

8.4.7 Related Work

OSNMT is related to explainable and interpretable NMT as well as alignment-based NMT. Both topics have been explored in Sec. 3.12.

The operation sequence model for SMT (Durrani et al., 2011, 2015) has been used in a number of MT evaluation systems (Durrani et al., 2016, 2014; Peter et al., 2016) and for post-editing (Pal et al., 2016), often in combination with a phrase-based model. The main difference to our OSNMT is that we have adapted the set of operations for neural models and are able to use it as stand-alone system, and not on top of a phrase-based system.

Our operation sequence model has some similarities with transition-based models used in other areas of NLP (Aharoni and Goldberg, 2017a; Dyer et al., 2015; Stenetorp, 2013). In particular, our `POP_SRC` operation is very similar to the *step* action of the hard alignment model of Aharoni and Goldberg (2017a). However, Aharoni and Goldberg (2017a) investigated

monotonic alignments for morphological inflections whereas we use a larger operation/action set to model complex word reorderings in machine translation.

8.4.8 Future Work

We have presented a way to use standard seq2seq models to generate a translation together with an alignment as linear sequence of operations. This greatly improves the interpretability of the model output as it establishes explicit alignment links between source and target tokens. However, the neural architecture we used here is representation-agnostic, i.e. we did not explicitly incorporate the alignments induced by an operation sequence into the neural model. For future work we are planning to adapt the Transformer model, for example by using positional embeddings of the source read head and the target write head in the Transformer attention layers.

8.5 Conclusion

In Ch. 4 we combined NMT with Hiero (Chiang, 2007), a hierarchical phrase-based machine translation approach. We followed up on this theme in this chapter and paired neural sequence models with various other kinds of hierarchical models. In Sec. 8.2 we reported on ensembling different linear representations of target-side syntactic parse trees for neural machine translation. We focused on text normalization in Sec. 8.3 – the conversion from written text such as abbreviations or numbers (e.g. "$123") into a a representation of its vocalization (e.g. "one hundred twenty three dollars"). We presented several context-sensitive neural architectures for text normalization that achieve good overall accuracy, but are prone to occasional unacceptable errors (e.g. "one thousand twenty three dollars"). We showed how constraining neural models with manually written covering grammars can help to reduce the risk of unacceptable errors. In Sec. 8.4 we introduced an operation sequence model for neural machine translation that represents the translation process as sequence of operations/actions. Such an operation sequence can be interpreted as a derivation in a formal grammar, with each operation corresponding to a rule in a multitext grammar.

Many discouraging hours will arise before the rainbow of accomplished goals will appear on the horizon.

Haile Selassie

9 Conclusion

The goal of this thesis was to present alternatives to the strict end-to-end deep learning paradigm that tends to view any NLP task from the same data mapping perspective, neglecting the special requirements tasks often have in practice. Most of our contributions aimed to break with this paradigm by either using language models or using hierarchical models. Using stand-alone language models contradicts the end-to-end philosophy as they are not trained jointly with the models they are paired with. Using explicit hierarchical structure conflicts with the end-to-end maxim of operating on input and output data only in the rawest form possible (e.g. plain text in machine translation). The following sections will review our contributions along these lines and point out possible future work.

9.1 Language Models

Ch. 4 focused on combining neural machine translation (NMT, Ch. 3) with statistical machine translation (SMT, Ch. 2) in hybrid systems. SMT uses a large number of features to score translations, including count-based or neural language models. Thus, our work on hybrid systems is an instance of using language models with NMT. We discussed several methods to combine SMT and NMT, varying in flexibility and runtime complexity. Syntactically guided neural machine translation (Sec. 4.3) constrains the NMT decoder to an SMT lattice

and is therefore fast but not very flexible. Our edit-distance based combination scheme in Sec. 4.4 is more flexible as it allows NMT to divert from the SMT search space, but it can be computationally challenging. The MBR-based scheme in Sec. 4.5 is both fast and flexible as it biases an unconstrained NMT decoder towards n-grams with high SMT score, and has been generalized to multiple system combination in Sec. 4.6. We argue that despite the success of NMT, monitoring the performance of hybrid systems is still important to guide future machine translation research. Why can NMT-SMT hybrids still improve over highly optimized pure NMT (Sec. 7.5) although SMT by its own lacks behind significantly? How can we use the decades of research on SMT to fix common pathologies of NMT such as hallucinations or non-words that did not affect SMT?

In Ch. 7 we explored the roles of language models in a more isolated way. In Sec. 7.2 we ordered words in a bag to form a fluent sentence using language models, demonstrating their usefulness for word reordering – one of the major challenges in machine translation. However, it is yet to be shown how our models and algorithms can be used to improve practical sequence generation tasks such as language generation or machine translation.

Sec. 7.3 proposed a cascade of FST composition operations to build up the search space for grammatical error correction, and then to rescore that search space with neural language models. Our language model based grammatical error correction approach is particularly suitable for adapting it to other languages in the future as it does not rely on large amounts of parallel training data.

We presented our *simple fusion* technique in Sec. 7.4 that improved NMT by integrating a pre-trained language model into the NMT training pipeline. While simple fusion seems to yield additive gains to back-translation on low-resource language pairs, it is yet to be seen whether the approach can be useful for large-scale MT as well.

Finally, we mixed sentence-level NMT with a document-level language model in Sec. 7.5 as a lightweight way to make the translations sensitive to cross-sentence context. Although our document-level language models achieved much better perplexity than sentence-level ones, they did not significantly improve the quality of the translations. Future experimentation has to be carried out to better understand the interactions between sentence-level translation models and document-level language models, and the role of cross-sentence context for translation in general.

9.2 Hierarchical Models

Most of our work on NMT-SMT hybrid systems in Ch. 4 was carried out with Hiero (Chiang, 2007), a *hierarchical* phrase-based statistical MT system. Thus, our review of Ch. 4 in the previous section also relates to the use of hierarchical models in neural machine translation.

Additionally, we devoted Ch. 8 to hierarchical models and how they can help neural sequence prediction. We reported in Sec. 8.2 that ensembling multiple representations of target-side syntax improves syntax-based NMT, but decoding can be challenging as syntax-based representations are usually long, and the decoder architecture for multi-representation ensembles is complex.

In Sec. 8.3 we followed up on prior work that constrained neural models for text normalization with covering grammars. The covering grammars were effective in reducing the risk of catastrophic errors, but they were written manually by a language expert in our experiments. Future work could try to reduce the human involvement in creating grammars. The initial experiments by Zhang et al. (2019a) on automatically learning such grammars from data could serve as a starting point.

Finally, we presented our work on operation sequence neural machine translation (OSNMT) in Sec. 8.4 that generates sequences of operations that describe the translation process rather than plain text. OSNMT has a clear link to hierarchical MT as its output can be interpreted as a derivation in a formal multitext grammar. OSNMT has the practical advantage over vanilla NMT of generating alignments together with the translations for better interpretability. In future work, the current operation set of OSNMT could be altered, for example to allow monotone translation on the target side, to reduce sequence lengths, or to be able to represent *n*:*m* alignments. The general concept of operation sequences could also be applicable to other sequence prediction tasks such as grammatical error correction or text normalization, possibly with a modified operation set.

9.3 Implementations

Ch. 6 developed a combined approach called *unfolding and shrinking* to efficient ensembling of neural models. The process of *unfolding* transforms an NMT ensemble into a single large neural network. *Shrinking* then reduces the size of the unfolded network to the size of one of the original NMT models while keeping the boost in translation performance from the ensembling. Unfolding and shrinking was motivated by the slow decoding speed of recurrent NMT models, and we therefore focused only on recurrent architectures. However, our research could be extended in the future to more recent architectures such as ConvS2S or the Transformer.

Another major contribution of this thesis is the SGNMT decoding framework (Ch. 5). The software architecture of SGNMT facilitates the transitioning to new NMT toolkits and the prototyping of new models and decoding strategies. It has become a valuable tool for teaching and research at the Cambridge MT group. SGNMT also features exact inference schemes that avoid search errors in neural decoding, and that can be used to analyze search errors and model errors in NMT. Surprisingly, we found that NMT prefers the empty translation under the absence of search errors for more than 50% of the sentences (Sec. 5.3), revealing a massive adequacy issue in NMT models. This finding calls for a stronger emphasis on adequacy in future NMT architectures.

9.4 Final Remarks

This thesis will not, and does not intend to put an end to the end-to-end paradigm a large part of the research in our field is following. We in fact acknowledge that this perspective has undoubtedly advanced our field substantially. However, we hope that the findings in this thesis encourage researchers to also pay attention to the nuances of the problems they are working on, and look beyond overall accuracies in well-controlled experiments as this is often crucial for valuable and practical contributions to our field.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., and Zheng, X. (2016). Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, Savannah, GA. USENIX Association.

Adel, H. and Schütze, H. (2017). Exploring different dimensions of attention for uncertainty detection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 22–34, Valencia, Spain. Association for Computational Linguistics.

Aharoni, R. and Goldberg, Y. (2017a). Morphological inflection generation with hard monotonic attention. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2004–2015, Vancouver, Canada. Association for Computational Linguistics.

Aharoni, R. and Goldberg, Y. (2017b). Towards string-to-tree neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 132–140, Vancouver, Canada. Association for Computational Linguistics.

Aharoni, R., Johnson, M., and Firat, O. (2019). Massively multilingual neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3874–3884, Minneapolis, Minnesota. Association for Computational Linguistics.

Ahmed, K., Keskar, N. S., and Socher, R. (2017). Weighted Transformer network for machine translation. *arXiv preprint arXiv:1711.02132*.

Aho, A. V. and Ullman, J. D. (1969). Syntax directed translations and the pushdown assembler. *Journal of Computer and System Sciences*, 3(1):37–56.

Akoury, N., Krishna, K., and Iyyer, M. (2019). Syntactically supervised Transformers for faster neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Florence, Italy. Association for Computational Linguistics.

Alishahi, A., Chrupala, G., and Linzen, T. (2019). Analyzing and interpreting neural networks for NLP: A report on the first BlackboxNLP workshop. *Natural Language Engineering*.

Alkhouli, T., Bretschner, G., and Ney, H. (2018). On the alignment problem in multi-head attention-based neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 177–185, Belgium, Brussels. Association for Computational Linguistics.

Alkhouli, T., Bretschner, G., Peter, J.-T., Hethnawi, M., Guta, A., and Ney, H. (2016). Alignment-based neural machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 54–65, Berlin, Germany. Association for Computational Linguistics.

Alkhouli, T. and Ney, H. (2017). Biasing attention-based recurrent neural networks using external alignment information. In *Proceedings of the Second Conference on Machine Translation*, pages 108–117, Copenhagen, Denmark. Association for Computational Linguistics.

Allauzen, C., Byrne, B., de Gispert, A., Iglesias, G., and Riley, M. D. (2014). Pushdown automata in statistical machine translation. *American Journal of Computational Linguistics*, 40(3):687–723.

Allauzen, C., Riley, M. D., Schalkwyk, J., Skut, W., and Mohri, M. (2007). OpenFst: A general and efficient weighted finite-state transducer library. In *Implementation and Application of Automata*, pages 11–23, Berlin, Heidelberg. Springer Berlin Heidelberg.

Alvarez-Melis, D. and Jaakkola, T. (2017). A causal framework for explaining the predictions of black-box sequence-to-sequence models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 412–421, Copenhagen, Denmark. Association for Computational Linguistics.

Álvaro Peris and Casacuberta, F. (2018). NMT-Keras: A very flexible toolkit with a focus on interactive NMT and online learning. *The Prague Bulletin of Mathematical Linguistics*, 111:113–124.

Andreas, J. and Klein, D. (2015). When and why are log-linear models self-normalizing? In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 244–249, Denver, Colorado. Association for Computational Linguistics.

Arase, Y. and Zhou, M. (2013). Machine translation detection from monolingual web-text. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1597–1607, Sofia, Bulgaria. Association for Computational Linguistics.

Artetxe, M., Labaka, G., and Agirre, E. (2017a). Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 451–462, Vancouver, Canada. Association for Computational Linguistics.

Artetxe, M., Labaka, G., Agirre, E., and Cho, K. (2017b). Unsupervised neural machine translation. *arXiv preprint arXiv:1710.11041*.

Arthur, P., Neubig, G., and Nakamura, S. (2016). Incorporating discrete translation lexicons into neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1557–1567, Austin, Texas. Association for Computational Linguistics.

Ataman, D., Negri, M., Turchi, M., and Federico, M. (2017). Linguistically motivated vocabulary reduction for neural machine translation from Turkish to English. *The Prague Bulletin of Mathematical Linguistics*, 108(1):331–342.

Atkinson, K. (2011). Automatically generated inflection database (AGID). http://wordlist.aspell.net/other/. [Online; accessed 24-December-2018].

Augasta, M. G. and Kathirvalavakumar, T. (2013). Pruning algorithms of neural networks — a comparative study. *Central European Journal of Computer Science*, 3(3):105–115.

Auli, M., Lopez, A., Hoang, H., and Koehn, P. (2009). A systematic analysis of translation model search spaces. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 224–232. Association for Computational Linguistics.

Avramidis, E., Macketanz, V., Burchardt, A., Helcl, J., and Uszkoreit, H. (2016). Deeper machine translation and evaluation for German. In *Proceedings of the 2nd Deep Machine Translation Workshop*, pages 29–38, Lisbon, Portugal. ÚFAL MFF UK.

Axelrod, A., He, X., and Gao, J. (2011). Domain adaptation via pseudo in-domain data selection. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 355–362, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

Ba, J. L., Mnih, V., and Kavukcuoglu, K. (2014). Multiple object recognition with visual attention. *arXiv preprint arXiv:1412.7755*.

Babaeizadeh, M., Smaragdis, P., and Campbell, R. H. (2016). NoiseOut: A simple way to prune neural networks. In *Proceedings of the 1st International Workshop on Efficient Methods for Deep Neural Networks (EMDNN)*.

Bach, N., Huang, F., and Al-Onaizan, Y. (2011). Goodness: A method for measuring machine translation confidence. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 211–219, Portland, Oregon, USA. Association for Computational Linguistics.

Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., and Samek, W. (2015). On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLOS ONE*, 10(7):1–46.

Bahar, P., Brix, C., and Ney, H. (2018). Towards two-dimensional sequence to sequence model in neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3009–3015, Brussels, Belgium. Association for Computational Linguistics.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *ICLR*.

Bangalore, S. and Riccardi, G. (2001). A finite-state approach to machine translation. In *Second Meeting of the North American Chapter of the Association for Computational Linguistics*.

Barrault, L., Bougares, F., Specia, L., Lala, C., Elliott, D., and Frank, S. (2018). Findings of the third shared task on multimodal machine translation. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 304–323, Belgium, Brussels. Association for Computational Linguistics.

Basho and Reichhold, J. (2013). *Basho: the complete haiku*. Kodansha International.

Bastien, F., Lamblin, P., Pascanu, R., Bergstra, J., Goodfellow, I., Bergeron, A., Bouchard, N., Warde-Farley, D., and Bengio, Y. (2012). Theano: New features and speed improvements. In *NIPS*.

Bastings, J., Aziz, W., Titov, I., and Sima'an, K. (2019). Modeling latent sentence structure in neural machine translation. *arXiv preprint arXiv:1901.06436*.

Bastings, J., Titov, I., Aziz, W., Marcheggiani, D., and Simaan, K. (2017). Graph convolutional encoders for syntax-aware neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1957–1967, Copenhagen, Denmark. Association for Computational Linguistics.

Bau, A., Belinkov, Y., Sajjad, H., Durrani, N., Dalvi, F., and Glass, J. (2018). Identifying and controlling important neurons in neural machine translation. *arXiv preprint arXiv:1811.01157*.

Bawden, R., Sennrich, R., Birch, A., and Haddow, B. (2018). Evaluating discourse phenomena in neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1304–1313, New Orleans, Louisiana. Association for Computational Linguistics.

Beck, D., de Gispert, A., Iglesias, G., Waite, A., and Byrne, B. (2016). Speed-constrained tuning for statistical machine translation using Bayesian optimization. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 856–865, San Diego, California. Association for Computational Linguistics.

Belinkov, Y. and Bisk, Y. (2017). Synthetic and natural noise both break neural machine translation. *arXiv preprint arXiv:1711.02173*.

Belinkov, Y., Durrani, N., Dalvi, F., Sajjad, H., and Glass, J. (2017). What do neural machine translation models learn about morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 861–872, Vancouver, Canada. Association for Computational Linguistics.

Bellegarda, J. R. (1997). A latent semantic analysis framework for large-span language modeling. In *Eurospeech*.

Bengio, S., Vinyals, O., Jaitly, N., and Shazeer, N. (2015). Scheduled sampling for sequence prediction with recurrent neural networks. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28*, pages 1171–1179. Curran Associates, Inc.

Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003). A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. (2009). Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 41–48. ACM.

Bengio, Y., Mesnil, G., Dauphin, Y. N., and Rifai, S. (2013). Better mixing via deep representations. In Dasgupta, S. and McAllester, D., editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 552–560, Atlanta, Georgia, USA. PMLR.

Bengio, Y., Schwenk, H., Senécal, J.-S., Morin, F., and Gauvain, J.-L. (2006). *Neural Probabilistic Language Models*, pages 137–186. Springer Berlin Heidelberg, Berlin, Heidelberg.

Bentivogli, L., Bisazza, A., Cettolo, M., and Federico, M. (2016). Neural versus phrase-based machine translation quality: A case study. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 257–267, Austin, Texas. Association for Computational Linguistics.

Bentivogli, L., Bisazza, A., Cettolo, M., and Federico, M. (2018). Neural versus phrase-based mt quality: An in-depth analysis on English–German and English–French. *Computer Speech & Language*, 49:52 – 70.

Blackwood, G., de Gispert, A., and Byrne, B. (2010). Efficient path counting transducers for minimum Bayes-risk decoding of statistical machine translation lattices. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 27–32, Uppsala, Sweden. Association for Computational Linguistics.

Bluche, T., Louradour, J., Knibbe, M., Moysset, B., Benzeghiba, M. F., and Kermorvant, C. (2014). The A2iA Arabic handwritten text recognition system at the Open HaRT2013 evaluation. In *2014 11th IAPR International Workshop on Document Analysis Systems*, pages 161–165. IEEE.

Bojar, O., Chatterjee, R., Federmann, C., Graham, Y., Haddow, B., Huang, S., Huck, M., Koehn, P., Liu, Q., Logacheva, V., Monz, C., Negri, M., Post, M., Rubino, R., Specia, L., and Turchi, M. (2017). Findings of the 2017 conference on machine translation (WMT17). In *Proceedings of the Second Conference on Machine Translation*, pages 169–214. Association for Computational Linguistics.

Bojar, O., Chatterjee, R., Federmann, C., Graham, Y., Haddow, B., Huck, M., Jimeno Yepes, A., Koehn, P., Logacheva, V., Monz, C., Negri, M., Neveol, A., Neves, M., Popel, M., Post, M., Rubino, R., Scarton, C., Specia, L., Turchi, M., Verspoor, K., and Zampieri, M. (2016). Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 131–198, Berlin, Germany. Association for Computational Linguistics.

Bojar, O. et al. (2019). Findings of the 2019 conference on machine translation (WMT19). In *Proceedings of the Fourth Conference on Machine Translation: Shared Task Papers*. Association for Computational Linguistics.

Bojar, O., Federmann, C., Fishel, M., Graham, Y., Haddow, B., Koehn, P., and Monz, C. (2018). Findings of the 2018 conference on machine translation (WMT18). In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 272–303. Association for Computational Linguistics.

Boulanger-Lewandowski, N., Bengio, Y., and Vincent, P. (2013). Audio chord recognition with recurrent neural networks. In *ISMIR*, pages 335–340. Citeseer.

Bourque, P., Fairley, R. E., et al. (2014). *Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0*. IEEE Computer Society Press.

Bowman, S., Pavlick, E., Grave, E., van Durme, B., Wang, A., Hula, J., Xia, P., Pappagari, R., McCoy, R. T., Patel, R., et al. (2018). Looking for ELMo's friends: Sentence-level pretraining beyond language modeling. *arXiv preprint arXiv:1812.10860*.

Bradbury, J. and Socher, R. (2017). Towards neural machine translation with latent tree attention. In *Proceedings of the 2nd Workshop on Structured Prediction for Natural Language Processing*, pages 12–16, Copenhagen, Denmark. Association for Computational Linguistics.

Brants, T., Popat, A. C., Xu, P., Och, F. J., and Dean, J. (2007). Large language models in machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 858–867, Prague, Czech Republic. Association for Computational Linguistics.

Britz, D., Le, Q. V., and Pryzant, R. (2017). Effective domain mixing for neural machine translation. In *Proceedings of the Second Conference on Machine Translation*, pages 118–126, Copenhagen, Denmark. Association for Computational Linguistics.

Brown, P. F., Cocke, J., Della Pietra, S. A., Della Pietra, V. J., Jelinek, F., Lafferty, J. D., Mercer, R. L., and Roossin, P. S. (1990). A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85.

Brown, P. F., Della Pietra, S. A., Della Pietra, V. J., and Mercer, R. L. (1993). The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

Brunner, G., Liu, Y., Pascual, D., Richter, O., and Wattenhofer, R. (2019). On the validity of self-attention as explanation in transformer models. *arXiv preprint arXiv:1908.04211*.

Bryant, C. and Briscoe, T. (2018). Language model based grammatical error correction without annotated training data. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 247–253, New Orleans, Louisiana. Association for Computational Linguistics.

Bryant, C., Felice, M., and Briscoe, T. (2017). Automatic annotation and evaluation of error types for grammatical error correction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 793–805. Association for Computational Linguistics.

Buciluǎ, C., Caruana, R., and Niculescu-Mizil, A. (2006). Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 535–541, New York, NY, USA. ACM.

Burlot, F. and Yvon, F. (2018). Using monolingual data in neural machine translation: A systematic study. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 144–155, Belgium, Brussels. Association for Computational Linguistics.

Byrne, B. (1993). Generalization and maximum likelihood from small data sets. In *Neural Networks for Signal Processing III - Proceedings of the 1993 IEEE-SP Workshop*, pages 197–206.

Caccia, M., Caccia, L., Fedus, W., Larochelle, H., Pineau, J., and Charlin, L. (2018). Language GANs falling short. *arXiv preprint arXiv:1811.02549*.

Calixto, I. and Liu, Q. (2019). An error analysis for image-based multi-modal neural machine translation. *Machine Translation*.

Carpuat, M., Vyas, Y., and Niu, X. (2017). Detecting cross-lingual semantic divergence for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 69–79, Vancouver. Association for Computational Linguistics.

Cashman, D., Patterson, G., Mosca, A., Watts, N., Robinson, S., and Chang, R. (2018). RNNbow: Visualizing learning via backpropagation gradients in RNNs. *IEEE Computer Graphics and Applications*, 38(6):39–50.

Castilho, S., Moorkens, J., Gaspari, F., Calixto, I., Tinsley, J., and Way, A. (2017a). Is neural machine translation the new state of the art? *The Prague Bulletin of Mathematical Linguistics*, 108(1):109–120.

Castilho, S., Moorkens, J., Gaspari, F., Sennrich, R., Sosoni, V., Georgakopoulou, P., Lohar, P., Way, A., Barone, A. V. M., and Gialama, M. (2017b). A comparative quality evaluation of PBSMT and NMT using professional translators. *Proceedings of Machine Translation Summit XVI, Nagoya, Japan*.

Cettolo, M., Federico, M., Bentivogli, L., Jan, N., Sebastian, S., Katsuitho, S., Koichiro, Y., and Christian, F. (2017). Overview of the IWSLT 2017 evaluation campaign. In *International Workshop on Spoken Language Translation*, pages 2–14.

Chan, W., Jaitly, N., Le, Q. V., and Vinyals, O. (2016). Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4960–4964.

Chandar, S., Ahn, S., Larochelle, H., Vincent, P., Tesauro, G., and Bengio, Y. (2016). Hierarchical memory networks. *arXiv preprint arXiv:1605.07427*.

Chelba, C., Mikolov, T., Schuster, M., Ge, Q., Brants, T., Koehn, P., and Robinson, T. (2014). One billion word benchmark for measuring progress in statistical language modeling. In *Fifteenth Annual Conference of the International Speech Communication Association (INTERSPEECH-2014)*, pages 2635–2639.

Chen, B., Cherry, C., Foster, G., and Larkin, S. (2017a). Cost weighting for neural machine translation domain adaptation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 40–46, Vancouver. Association for Computational Linguistics.

Chen, H., Huang, S., Chiang, D., and Chen, J. (2017b). Improved neural machine translation with a syntax-aware encoder and decoder. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1936–1945, Vancouver, Canada. Association for Computational Linguistics.

Chen, K., Wang, R., Utiyama, M., Liu, L., Tamura, A., Sumita, E., and Zhao, T. (2017c). Neural machine translation with source dependency representation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2846–2852, Copenhagen, Denmark. Association for Computational Linguistics.

Chen, K., Wang, R., Utiyama, M., Sumita, E., and Zhao, T. (2018a). Syntax-directed attention for neural machine translation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Chen, M. X., Firat, O., Bapna, A., Johnson, M., Macherey, W., Foster, G., Jones, L., Schuster, M., Shazeer, N., Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, Ł., Chen, Z., Wu, Y., and Hughes, M. (2018b). The best of both worlds: Combining recent advances in neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 76–86, Melbourne, Australia. Association for Computational Linguistics.

Chen, W., Matusov, E., Khadivi, S., and Peter, J.-T. (2016). Guided alignment training for topic-aware neural machine translation. *AMTA 2016, Vol.*, page 121.

Chen, Y., Gilroy, S., Maletti, A., May, J., and Knight, K. (2018c). Recurrent neural networks as weighted language recognizers. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2261–2271, New Orleans, Louisiana. Association for Computational Linguistics.

Chen, Y., Li, V. O., Cho, K., and Bowman, S. (2018d). A stable and effective learning strategy for trainable greedy decoding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 380–390, Brussels, Belgium. Association for Computational Linguistics.

Chen, Z., Luitjens, J., Xu, H., Wang, Y., Povey, D., and Khudanpur, S. (2018e). A GPU-based WFST decoder with exact lattice generation. In *Proc. Interspeech 2018*, pages 2212–2216.

Cheng, J., Dong, L., and Lapata, M. (2016a). Long short-term memory-networks for machine reading. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 551–561, Austin, Texas. Association for Computational Linguistics.

Cheng, Y., Shen, S., He, Z., He, W., Wu, H., Sun, M., and Liu, Y. (2016b). Agreement-based joint training for bidirectional attention-based neural machine translation. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 2761–2767. AAAI Press.

Cheng, Y., Tu, Z., Meng, F., Zhai, J., and Liu, Y. (2018). Towards robust neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1756–1766, Melbourne, Australia. Association for Computational Linguistics.

Cheng, Y., Xu, W., He, Z., He, W., Wu, H., Sun, M., and Liu, Y. (2016c). Semi-supervised learning for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1965–1974, Berlin, Germany. Association for Computational Linguistics.

Cheng, Y., Yang, Q., Liu, Y., Sun, M., and Xu, W. (2017). Joint training for pivot-based neural machine translation. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, pages 3974–3980. AAAI Press.

Cherry, C., Foster, G., Bapna, A., Firat, O., and Macherey, W. (2018). Revisiting character-based neural machine translation with capacity and compression. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4295–4305, Brussels, Belgium. Association for Computational Linguistics.

Chiang, D. (2005). A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 263–270, Ann Arbor, Michigan. Association for Computational Linguistics.

Chiang, D. (2007). Hierarchical phrase-based translation. *American Journal of Computational Linguistics*, 33(2):201–228.

Chiang, D., Knight, K., and Wang, W. (2009). 11,001 new features for statistical machine translation. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 218–226, Boulder, Colorado. Association for Computational Linguistics.

Chitnis, R. and DeNero, J. (2015). Variable-length word encodings for neural translation models. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2088–2093, Lisbon, Portugal. Association for Computational Linguistics.

Cho, K. (2016). Noisy parallel approximate decoding for conditional recurrent language model. *arXiv preprint arXiv:1605.03835*.

Cho, K. and Esipova, M. (2016). Can neural machine translation do simultaneous translation? *arXiv preprint arXiv:1606.02012*.

Cho, K., van Merrienboer, B., Bahdanau, D., and Bengio, Y. (2014a). On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar. Association for Computational Linguistics.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014b). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Choi, E., Hewlett, D., Uszkoreit, J., Polosukhin, I., Lacoste, A., and Berant, J. (2017a). Coarse-to-fine question answering for long documents. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 209–220, Vancouver, Canada. Association for Computational Linguistics.

Choi, G.-H., Shin, J.-H., and Kim, Y.-K. (2018a). Improving a multi-source neural machine translation model with corpus extension for low-resource languages. In *Proceedings of the 11th Language Resources and Evaluation Conference*, Miyazaki, Japan. European Language Resource Association.

Choi, H., Cho, K., and Bengio, Y. (2017b). Context-dependent word representation for neural machine translation. *Computer Speech & Language*, 45:149 – 160.

Choi, H., Cho, K., and Bengio, Y. (2018b). Fine-grained attention mechanism for neural machine translation. *Neurocomputing*, 284:171 – 176.

Chollampatt, S. and Ng, H. T. (2017). Connecting the dots: Towards human-level grammatical error correction. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 327–333, Copenhagen, Denmark. Association for Computational Linguistics.

Chollampatt, S. and Ng, H. T. (2018). A multilayer convolutional encoder-decoder neural network for grammatical error correction. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, New Orleans, Louisiana, USA.

Chollampatt, S., Taghipour, K., and Ng, H. T. (2016). Neural network translation models for grammatical error correction. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 2768–2774. AAAI Press.

Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1800–1807. IEEE.

Chomsky, N. (1957). *Syntactic Structures*. Mouton and Co., The Hague.

Chorowski, J., Bahdanau, D., Cho, K., and Bengio, Y. (2014). End-to-end continuous speech recognition using attention-based recurrent NN: First results. In *NIPS 2014 Workshop on Deep Learning, December 2014*.

Chorowski, J. and Jaitly, N. (2017). Towards better decoding and language model integration in sequence to sequence models. In *Proc. Interspeech 2017*, pages 523–527.

Chu, C., Dabre, R., and Kurohashi, S. (2018). A comprehensive empirical comparison of domain adaptation methods for neural machine translation. *Journal of Information Processing*, 26:529–538.

Chu, C. and Wang, R. (2018). A survey of domain adaptation for neural machine translation. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1304–1319, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Chung, J., Cho, K., and Bengio, Y. (2016). A character-level decoder without explicit segmentation for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1693–1703, Berlin, Germany. Association for Computational Linguistics.

Cohn, T., Hoang, C. D. V., Vymolova, E., Yao, K., Dyer, C., and Haffari, G. (2016). Incorporating structural alignment biases into an attentional neural translation model. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 876–885, San Diego, California. Association for Computational Linguistics.

Collobert, R., Hannun, A., and Synnaeve, G. (2019). A fully differentiable beam search decoder. *arXiv preprint arXiv:1902.06022*.

Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 160–167, New York, NY, USA. ACM.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537.

Conneau, A., Kiela, D., Schwenk, H., Barrault, L., and Bordes, A. (2017a). Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680. Association for Computational Linguistics.

Conneau, A., Kruszewski, G., Lample, G., Barrault, L., and Baroni, M. (2018). What you can cram into a single vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136. Association for Computational Linguistics.

Conneau, A., Lample, G., Ranzato, M., Denoyer, L., and Jégou, H. (2017b). Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Cormen, T. H. (2009). *Introduction to algorithms*. MIT press.

Costa-jussà, M. R., Escolano, C., and Fonollosa, J. A. (2017). Byte-based neural machine translation. In *Proceedings of the First Workshop on Subword and Character Level Models in NLP*, pages 154–158, Copenhagen, Denmark. Association for Computational Linguistics.

Costa-jussà, M. R. and Fonollosa, J. A. (2016). Character-based neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 357–361, Berlin, Germany. Association for Computational Linguistics.

Crego, J., Kim, J., Klein, G., Rebollo, A., Yang, K., Senellart, J., Akhanov, E., Brunelle, P., Coquard, A., Deng, Y., et al. (2016). SYSTRAN's pure neural machine translation systems. *arXiv preprint arXiv:1610.05540*.

Cromieres, F., Chu, C., Nakazawa, T., and Kurohashi, S. (2016). Kyoto university participation to WAT 2016. In *Proceedings of the 3rd Workshop on Asian Translation (WAT2016)*, pages 166–174, Osaka, Japan. The COLING 2016 Organizing Committee.

Cromieres, F., Nakazawa, T., and Dabre, R. (2017). Neural machine translation: Basics, practical aspects and recent trends. In *Proceedings of the IJCNLP 2017, Tutorial Abstracts*, pages 11–13, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Cui, L., Zhang, D., Liu, S., Li, M., and Zhou, M. (2013). Bilingual data cleaning for SMT using graph-based random walk. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 340–345, Sofia, Bulgaria. Association for Computational Linguistics.

Currey, A. and Heafield, K. (2018). Multi-source syntactic neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2961–2966, Brussels, Belgium. Association for Computational Linguistics.

Currey, A., Miceli Barone, A. V., and Heafield, K. (2017). Copied monolingual data improves low-resource neural machine translation. In *Proceedings of the Second Conference on Machine Translation*, pages 148–156, Copenhagen, Denmark. Association for Computational Linguistics.

Dabre, R., Chu, C., and Kunchukuttan, A. (2019). A survey of multilingual neural machine translation. *arXiv preprint arXiv:1905.05395*.

Dabre, R., Nakagawa, T., and Kazawa, H. (2017). An empirical study of language relatedness for transfer learning in neural machine translation. In *Proceedings of the 31st Pacific Asia Conference on Language, Information and Computation*, pages 282–286. The National University (Phillippines).

Dahlmann, L., Matusov, E., Petrushkov, P., and Khadivi, S. (2017). Neural machine translation leveraging phrase-based models in a hybrid search. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1411–1420, Copenhagen, Denmark. Association for Computational Linguistics.

Dahlmeier, D. and Ng, H. T. (2012). Better evaluation for grammatical error correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 568–572. Association for Computational Linguistics.

Dahlmeier, D., Ng, H. T., and Wu, S. M. (2013). Building a large annotated corpus of learner English: The NUS corpus of learner English. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 22–31. Association for Computational Linguistics.

Dai, Z., Yang, Z., Yang, Y., Cohen, W. W., Carbonell, J., Le, Q. V., and Salakhutdinov, R. (2019). Transformer-XL: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.

Dakwale, P. and Monz, C. (2017). Fine-tuning for neural machine translation with limited degradation across in-and out-of-domain data. *Proceedings of the XVI Machine Translation Summit*, page 117.

Dalvi, F., Durrani, N., Sajjad, H., Belinkov, Y., Bau, A., and Glass, J. (2019). What is one grain of sand in the desert? Analyzing individual neurons in deep NLP models. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Dalvi, F., Nortonsmith, A., Bau, A., Belinkov, Y., Sajjad, H., Durrani, N., and Glass, J. (2018). NeuroX: A toolkit for analyzing individual neurons in neural networks. *arXiv preprint arXiv:1812.09359*.

Daniil, G., Kalaidin, P., and Malykh, V. (2019). Self-attentive model for headline generation. *arXiv preprint arXiv:1901.07786*.

de Gispert, A., Blackwood, G., Iglesias, G., and Byrne, B. (2013). N-gram posterior probability confidence measures for statistical machine translation: An empirical study. *Machine Translation*, 27(2):85–114.

de Gispert, A., Iglesias, G., Blackwood, G., Banga, E. R., and Byrne, B. (2010). Hierarchical phrase-based translation with weighted finite-state transducers and shallow-n grammars. *American Journal of Computational Linguistics*, 36(3):505–533.

de Gispert, A., Tomalin, M., and Byrne, B. (2014). Word ordering with phrase-based grammars. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 259–268, Gothenburg, Sweden. Association for Computational Linguistics.

de Gispert, A., Virpioja, S., Kurimo, M., and Byrne, B. (2009). Minimum Bayes risk combination of translation hypotheses from alternative morphological decompositions. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 73–76, Boulder, Colorado. Association for Computational Linguistics.

Denil, M., Shakibi, B., Dinh, L., Ranzato, M., and de Freitas, N. (2013). Predicting parameters in deep learning. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 26*, pages 2148–2156. Curran Associates, Inc.

Denton, E. L., Zaremba, W., Bruna, J., LeCun, Y., and Fergus, R. (2014). Exploiting linear structure within convolutional networks for efficient evaluation. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 1269–1277. Curran Associates, Inc.

Devlin, J. (2017). Sharp models on dull hardware: Fast and accurate neural machine translation decoding on the CPU. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2820–2825, Copenhagen, Denmark. Association for Computational Linguistics.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional Transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Devlin, J., Zbib, R., Huang, Z., Lamar, T., Schwartz, R., and Makhoul, J. (2014). Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1370–1380, Baltimore, Maryland. Association for Computational Linguistics.

Di Gangi, M. A. and Federico, M. (2018). Deep neural machine translation with weakly-recurrent units. In *Proceedings of the 21st Annual Conference of the European Association for Machine Translation: 28-30 May 2018, Universitat d'Alacant, Alacant, Spain*, pages 119–128. European Association for Machine Translation.

Dietterich, T. G. (2000). Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15, Berlin, Heidelberg. Springer Berlin Heidelberg.

Ding, Y., Liu, Y., Luan, H., and Sun, M. (2017). Visualizing and understanding neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1150–1159, Vancouver, Canada. Association for Computational Linguistics.

Domhan, T. (2018). How much attention do you need? A granular analysis of neural machine translation architectures. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1799–1808, Melbourne, Australia. Association for Computational Linguistics.

Domhan, T. and Hieber, F. (2017). Using target-side monolingual data for neural machine translation through multi-task learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1500–1505, Copenhagen, Denmark. Association for Computational Linguistics.

Domingo, M., Garcıa-Martınez, M., Helle, A., and Casacuberta, F. (2018). How much does tokenization affect in neural machine translation? *arXiv preprint arXiv:1812.08621*.

Dong, D., Wu, H., He, W., Yu, D., and Wang, H. (2015). Multi-task learning for multiple language translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1723–1732, Beijing, China. Association for Computational Linguistics.

Dong, L. and Lapata, M. (2016). Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33–43, Berlin, Germany. Association for Computational Linguistics.

dos Santos, C. and Gatti, M. (2014). Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 69–78. Dublin City University and Association for Computational Linguistics.

Doshi-Velez, F. and Kim, B. (2017). Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*.

Dowling, M., Lynn, T., Poncelas, A., and Way, A. (2018). SMT versus NMT: Preliminary comparisons for irish. In *Proceedings of the AMTA 2018 Workshop on Technologies for MT of Low Resource Languages (LoResMT 2018)*, pages 12–20, Boston, MA. Association for Machine Translation in the Americas.

Du, J. and Way, A. (2017). Neural pre-translation for hybrid machine translation. In *Proceedings of MT Summit*, volume 16, pages 27–40.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, pages 2121–2159.

Duong, L., Anastasopoulos, A., Chiang, D., Bird, S., and Cohn, T. (2016). An attentional model for speech translation without transcription. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 949–959, San Diego, California. Association for Computational Linguistics.

Durrani, N., Dalvi, F., Sajjad, H., Belinkov, Y., and Nakov, P. (2018). What is in a translation unit? Comparing character and subword representations beyond translation. *openreview.net*.

Durrani, N., Dalvi, F., Sajjad, H., and Vogel, S. (2016). QCRI machine translation systems for IWSLT 16. In *International Workshop on Spoken Language Translation. Seattle, WA, USA*.

Durrani, N., Haddow, B., Koehn, P., and Heafield, K. (2014). Edinburgh's phrase-based machine translation systems for WMT-14. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 97–104, Baltimore, Maryland, USA. Association for Computational Linguistics.

Durrani, N., Schmid, H., and Fraser, A. (2011). A joint sequence translation model with integrated reordering. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1045–1054, Portland, Oregon, USA. Association for Computational Linguistics.

Durrani, N., Schmid, H., Fraser, A., Koehn, P., and Schütze, H. (2015). The operation sequence model—combining n-gram-based and phrase-based statistical machine translation. *Computational Linguistics*, 41(2):157–186.

Dyer, C. (2014). Notes on noise contrastive estimation and negative sampling. *arXiv preprint arXiv:1410.8251*.

Dyer, C., Ballesteros, M., Ling, W., Matthews, A., and Smith, N. A. (2015). Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343, Beijing, China. Association for Computational Linguistics.

Dyer, C., Kuncoro, A., Ballesteros, M., and Smith, N. A. (2016). Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209, San Diego, California. Association for Computational Linguistics.

Edunov, S., Ott, M., Auli, M., and Grangier, D. (2018a). Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 489–500, Brussels, Belgium. Association for Computational Linguistics.

Edunov, S., Ott, M., Auli, M., Grangier, D., and Ranzato, M. (2018b). Classical structured prediction losses for sequence to sequence learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 355–364, New Orleans, Louisiana. Association for Computational Linguistics.

Elliott, D., Frank, S., Barrault, L., Bougares, F., and Specia, L. (2017). Findings of the second shared task on multimodal machine translation and multilingual image description. In *Proceedings of the Second Conference on Machine Translation*, pages 215–233, Copenhagen, Denmark. Association for Computational Linguistics.

Elliott, D., Frank, S., and Hasler, E. (2015). Multilingual image description with neural sequence models. *arXiv preprint arXiv:1510.04709*.

ElMaghraby, A. and Rafea, A. (2019). Enhancing translation from English to Arabic using two-phase decoder translation. In Arai, K., Kapoor, S., and Bhatia, R., editors, *Intelligent Systems and Applications*, pages 539–549, Cham. Springer International Publishing.

Elman, J. L. (1990). Finding structure in time. *Cognitive science*, 14(2):179–211.

Er, M. J., Zhang, Y., Wang, N., and Pratama, M. (2016). Attention pooling-based convolutional neural network for sentence modelling. *Information Sciences*, 373:388 – 403.

Eriguchi, A., Hashimoto, K., and Tsuruoka, Y. (2016). Tree-to-sequence attentional neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 823–833, Berlin, Germany. Association for Computational Linguistics.

Eriguchi, A., Hashimoto, K., and Tsuruoka, Y. (2019). Incorporating source-side phrase structures into neural machine translation. *Computational Linguistics*, 45(2):267–292.

Eriguchi, A., Tsuruoka, Y., and Cho, K. (2017). Learning to parse and translate improves neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 72–78, Vancouver, Canada. Association for Computational Linguistics.

Escolano, C., Costa-jussà, M. R., and Fonollosa, J. A. (2018). (self-attentive) autoencoder-based universal language representation for machine translation. *arXiv preprint arXiv:1810.06351*.

Fan, A., Grangier, D., and Auli, M. (2018). Controllable abstractive summarization. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 45–54, Melbourne, Australia. Association for Computational Linguistics.

Feng, J., Kong, L., Huang, P.-S., Wang, C., Huang, D., Mao, J., Qiao, K., and Zhou, D. (2018a). Neural phrase-to-phrase machine translation. *arXiv preprint arXiv:1811.02172*.

Feng, S., Liu, S., Li, M., and Zhou, M. (2016). Implicit distortion and fertility models for attention-based encoder-decoder NMT model. *arXiv preprint arXiv:1601.03317*.

Feng, S., Wallace, E., Grissom II, A., Iyyer, M., Rodriguez, P., and Boyd-Graber, J. (2018b). Pathologies of neural models make interpretations difficult. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3719–3728, Brussels, Belgium. Association for Computational Linguistics.

Feng, Y., Zhang, S., Zhang, A., Wang, D., and Abel, A. (2017). Memory-augmented neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1390–1399, Copenhagen, Denmark. Association for Computational Linguistics.

Firat, O., Cho, K., and Bengio, Y. (2016a). Multi-way, multilingual neural machine translation with a shared attention mechanism. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 866–875, San Diego, California. Association for Computational Linguistics.

Firat, O., Cho, K., Sankaran, B., Vural, F. T. Y., and Bengio, Y. (2017). Multi-way, multilingual neural machine translation. *Computer Speech & Language*, 45:236 – 252.

Firat, O., Sankaran, B., Al-Onaizan, Y., Yarman Vural, F. T., and Cho, K. (2016b). Zero-resource translation with multi-lingual neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 268–277, Austin, Texas. Association for Computational Linguistics.

Fiscus, J. G. (1997). A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (ROVER). In *1997 IEEE Workshop on Automatic Speech Recognition and Understanding Proceedings*, pages 347–354.

Foster, G., Goutte, C., and Kuhn, R. (2010). Discriminative instance weighting for domain adaptation in statistical machine translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 451–459, Cambridge, MA. Association for Computational Linguistics.

Frederking, R., Nirenburg, S., Farwell, D., Helmreich, S., Hovy, E., Knight, K., Beale, S., Domashnev, C., Attardo, D., Grannes, D., et al. (1994). Integrating translations from multiple sources within the Pangloss Mark III machine translation system. In *AMTA*, pages 73–80.

Freitag, M. and Al-Onaizan, Y. (2016). Fast domain adaptation for neural machine translation. *arXiv preprint arXiv:1612.06897*.

Freitag, M. and Al-Onaizan, Y. (2017). Beam search strategies for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 56–60, Vancouver. Association for Computational Linguistics.

Freitag, M., Al-Onaizan, Y., and Sankaran, B. (2017). Ensemble distillation for neural machine translation. *arXiv preprint arXiv:1702.01802*.

French, R. M. (1999). Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128 – 135.

Fügen, C., Waibel, A., and Kolss, M. (2007). Simultaneous translation of lectures and speeches. *Machine translation*, 21(4):209–252.

Gage, P. (1994). A new algorithm for data compression. *The C Users Journal*, 12(2):23–38.

Gao, J. (2016). Variable length word encodings for neural translation models. MPhil dissertation, University of Cambridge.

García-Martínez, M., Barrault, L., and Bougares, F. (2016). Factored neural machine translation architectures. In *International Workshop on Spoken Language Translation (IWSLT'16)*.

García-Martínez, M., Barrault, L., and Bougares, F. (2017). Neural machine translation by generating multiple linguistic factors. In Camelin, N., Estève, Y., and Martín-Vide, C., editors, *Statistical Language and Speech Processing*, pages 21–31, Cham. Springer International Publishing.

Ge, T., Wei, F., and Zhou, M. (2018a). Fluency boost learning and inference for neural grammatical error correction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1055–1065, Melbourne, Australia. Association for Computational Linguistics.

Ge, T., Wei, F., and Zhou, M. (2018b). Reaching human-level performance in automatic grammatical error correction: An empirical study. *arXiv preprint arXiv:1807.01270*.

Gehring, J., Auli, M., Grangier, D., and Dauphin, Y. N. (2017a). A convolutional encoder model for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 123–135, Vancouver, Canada. Association for Computational Linguistics.

Gehring, J., Auli, M., Grangier, D., Yarats, D., and Dauphin, Y. N. (2017b). Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 1243–1252. JMLR.org.

Geng, X., Feng, X., Qin, B., and Liu, T. (2018). Adaptive multi-pass decoder for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 523–532, Brussels, Belgium. Association for Computational Linguistics.

Ghader, H. and Monz, C. (2017). What does attention in neural machine translation pay attention to? In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 30–39, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Gimpel, K., Batra, D., Dyer, C., and Shakhnarovich, G. (2013). A systematic exploration of diversity in machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1100–1111, Seattle, Washington, USA. Association for Computational Linguistics.

Goel, V., Kumar, S., and Byrne, B. (2000). Segmental minimum Bayes-risk ASR voting strategies. In *Interspeech*, pages 139–142.

Goldberg, Y. (2016). A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, 57:345–420.

Goldberg, Y. (2019). Assessing BERT's syntactic abilities. *arXiv preprint arXiv:1901.05287*.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Goodfellow, I., Mirza, M., Xiao, D., Courville, A., and Bengio, Y. (2013a). An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc.

Goodfellow, I., Warde-farley, D., Mirza, M., Courville, A., and Bengio, Y. (2013b). Maxout networks. In *ICML*, pages 1319–1327.

Goyal, K., Neubig, G., Dyer, C., and Berg-Kirkpatrick, T. (2018). A continuous relaxation of beam search for end-to-end training of neural sequence models. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Graves, A. and Jaitly, N. (2014). Towards end-to-end speech recognition with recurrent neural networks. In *International conference on machine learning*, pages 1764–1772.

Graves, A., Wayne, G., and Danihelka, I. (2014). Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwińska, A., Colmenarejo, S. G., Grefenstette, E., Ramalho, T., Agapiou, J., et al. (2016). Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471.

Grefenstette, E., Hermann, K. M., Suleyman, M., and Blunsom, P. (2015). Learning to transduce with unbounded memory. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28*, pages 1828–1836. Curran Associates, Inc.

Grissom II, A., He, H., Boyd-Graber, J., Morgan, J., and Daumé III, H. (2014). Don't until the final verb wait: Reinforcement learning for simultaneous machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1342–1352, Doha, Qatar. Association for Computational Linguistics.

Grundkiewicz, R. and Junczys-Dowmunt, M. (2018). Near human-level performance in grammatical error correction with hybrid machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 284–290, New Orleans, Louisiana. Association for Computational Linguistics.

Gu, J., Bradbury, J., Xiong, C., Li, V. O., and Socher, R. (2017a). Non-autoregressive neural machine translation. *arXiv preprint arXiv:1711.02281*.

Gu, J., Cho, K., and Li, V. O. (2017b). Trainable greedy decoding for neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1968–1978, Copenhagen, Denmark. Association for Computational Linguistics.

Gu, J., Liu, Q., and Cho, K. (2019a). Insertion-based decoding with automatically inferred generation order. *arXiv preprint arXiv:1902.01370*.

Gu, J., Neubig, G., Cho, K., and Li, V. O. (2017c). Learning to translate in real-time with neural machine translation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1053–1062, Valencia, Spain. Association for Computational Linguistics.

Gu, J., Wang, C., and Zhao, J. (2019b). Levenshtein Transformer. *arXiv preprint arXiv:1905.11006*.

Gulcehre, C., Ahn, S., Nallapati, R., Zhou, B., and Bengio, Y. (2016). Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 140–149, Berlin, Germany. Association for Computational Linguistics.

Gulcehre, C., Dutil, F., Trischler, A., and Bengio, Y. (2017a). Plan, attend, generate: Character-level neural machine translation with planning. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 228–234, Vancouver, Canada. Association for Computational Linguistics.

Gulcehre, C., Firat, O., Xu, K., Cho, K., Barrault, L., Lin, H.-C., Bougares, F., Schwenk, H., and Bengio, Y. (2015). On using monolingual corpora in neural machine translation. *arXiv preprint arXiv:1503.03535*.

Gulcehre, C., Firat, O., Xu, K., Cho, K., and Bengio, Y. (2017b). On integrating a language model into neural machine translation. *Computer Speech & Language*, 45:137 – 148.

Guo, J., Tan, X., He, D., Qin, T., Xu, L., and Liu, T.-Y. (2018). Non-autoregressive neural machine translation with enhanced decoder input. *arXiv preprint arXiv:1812.09664*.

Guo, Q., Qiu, X., Liu, P., Shao, Y., Xue, X., and Zhang, Z. (2019). Star-Transformer. *arXiv preprint arXiv:1902.09113*.

Gutmann, M. and Hyvärinen, A. (2010). Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In Teh, Y. W. and Titterington, M., editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 297–304, Chia Laguna Resort, Sardinia, Italy. PMLR.

Ha, T.-L., Niehues, J., and Waibel, A. (2016). Toward multilingual neural machine translation with universal encoder and decoder. In *International Workshop on Spoken Language Translation IWSLT*.

Ha, T.-L., Niehues, J., and Waibel, A. (2017). Effective strategies in zero-shot neural machine translation. In *International Workshop on Spoken Language Translation*.

Han, S., Pool, J., Tran, J., and Dally, W. (2015). Learning both weights and connections for efficient neural network. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28*, pages 1135–1143. Curran Associates, Inc.

Hans, K. and Milton, R. (2016). Improving the performance of neural machine translation involving morphologically rich languages. *arXiv preprint arXiv:1612.02482*.

Hansen, L. K. and Salamon, P. (1990). Neural network ensembles. *IEEE transactions on pattern analysis and machine intelligence*, 12(10):993–1001.

Hao, J., Wang, X., Yang, B., Wang, L., Zhang, J., and Tu, Z. (2019). Modeling recurrence for Transformer. *arXiv preprint arXiv:1904.03092*.

Hasler, E., de Gispert, A., Iglesias, G., and Byrne, B. (2018). Neural machine translation decoding with terminology constraints. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 506–512, New Orleans, Louisiana. Association for Computational Linguistics.

Hasler, E., de Gispert, A., Stahlberg, F., Waite, A., and Byrne, B. (2017a). Source sentence simplification for statistical machine translation. *Computer Speech & Language*, 45:221 – 235.

Hasler, E., Stahlberg, F., Tomalin, M., de Gispert, A., and Byrne, B. (2017b). A comparison of neural models for word ordering. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 208–212, Santiago de Compostela, Spain. Association for Computational Linguistics.

Hassan, H., Aue, A., Chen, C., Chowdhary, V., Clark, J. H., Federmann, C., Huang, X., Junczys-Dowmunt, M., Lewis, W. D., Li, M., et al. (2018). Achieving human parity on automatic Chinese to English news translation. *arXiv preprint arXiv:1803.05567*.

Hassibi, B., Stork, D. G., et al. (1993). Second order derivatives for network pruning: Optimal brain surgeon. *Advances in neural information processing systems*, pages 164–171.

He, D., Lu, H., Xia, Y., Qin, T., Wang, L., and Liu, T.-Y. (2017). Decoding with value networks for neural machine translation. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 178–187. Curran Associates, Inc.

He, D., Xia, Y., Qin, T., Wang, L., Yu, N., Liu, T.-Y., and Ma, W.-Y. (2016a). Dual learning for machine translation. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems 29*, pages 820–828. Curran Associates, Inc.

He, H., Boyd-Graber, J., and Daumé III, H. (2016b). Interpretese vs. Translationese: The uniqueness of human strategies in simultaneous interpretation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 971–976, San Diego, California. Association for Computational Linguistics.

He, K., Zhang, X., Ren, S., and Sun, J. (2016c). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

He, T., Tan, X., Xia, Y., He, D., Qin, T., Chen, Z., and Liu, T.-Y. (2018a). Layer-wise coordination between encoder and decoder for neural machine translation. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31*, pages 7944–7954. Curran Associates, Inc.

He, W., He, Z., Wu, H., and Wang, H. (2016d). Improved neural machine translation with SMT features. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 151–157. AAAI Press.

He, X., Haffari, G., and Norouzi, M. (2018b). Sequence to sequence mixture model for diverse machine translation. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 583–592, Brussels, Belgium. Association for Computational Linguistics.

Heafield, K. (2011). KenLM: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland. Association for Computational Linguistics.

Heafield, K., Pouzyrevsky, I., Clark, J. H., and Koehn, P. (2013). Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 690–696, Sofia, Bulgaria. Association for Computational Linguistics.

Hebb, D. (1949). *The Organization of Behavior*. Wiley.

Helcl, J. and Libovický, J. (2017). Neural Monkey: An open-source tool for sequence learning. *The Prague Bulletin of Mathematical Linguistics*, pages 5–17.

Hermann, K. M., Kocisky, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., and Blunsom, P. (2015). Teaching machines to read and comprehend. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28*, pages 1693–1701. Curran Associates, Inc.

Hieber, F., Domhan, T., Denkowski, M., Vilar, D., Sokolov, A., Clifton, A., and Post, M. (2017). Sockeye: A toolkit for neural machine translation. *arXiv preprint arXiv:1712.05690*.

Hildebrand, A. S., Eck, M., Vogel, S., and Waibel, A. (2005). Adaptation of the translation model for statistical machine translation based on information retrieval. In *Proceedings of EAMT*, volume 2005, pages 133–142.

Hinton, G. E., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*.

Hitschler, J., Schamoni, S., and Riezler, S. (2016). Multimodal pivots for image caption translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2399–2409, Berlin, Germany. Association for Computational Linguistics.

Hoang, H., Dwojak, T., Krislauks, R., Torregrosa, D., and Heafield, K. (2018a). Fast neural machine translation implementation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 116–121, Melbourne, Australia. Association for Computational Linguistics.

Hoang, V. C. D., Haffari, G., and Cohn, T. (2017). Towards decoding as continuous optimisation in neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 146–156, Copenhagen, Denmark. Association for Computational Linguistics.

Hoang, V. C. D., Koehn, P., Haffari, G., and Cohn, T. (2018b). Iterative back-translation for neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 18–24, Melbourne, Australia. Association for Computational Linguistics.

Hochreiter, S., Bengio, Y., Frasconi, P., and Schmidhuber, J. (2001). Gradient flow in recurrent nets: The difficulty of learning long-term dependencies.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Hoshen, Y. and Wolf, L. (2018). Non-adversarial unsupervised word translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 469–478, Brussels, Belgium. Association for Computational Linguistics.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

Hu, M., Peng, Y., Huang, Z., Qiu, X., Wei, F., and Zhou, M. (2018). Reinforced mnemonic reader for machine reading comprehension. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, pages 4099–4106. AAAI Press.

Huang, L., Zhao, K., and Ma, M. (2017a). When to finish? Optimal beam search for neural text generation (modulo beam size). In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2134–2139, Copenhagen, Denmark. Association for Computational Linguistics.

Huang, P.-S., Wang, C., Huang, S., Zhou, D., and Deng, L. (2017b). Towards neural phrase-based machine translation. *arXiv preprint arXiv:1706.05565*.

Huang, P.-Y., Liu, F., Shiang, S.-R., Oh, J., and Dyer, C. (2016). Attention-based multimodal neural machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 639–645, Berlin, Germany. Association for Computational Linguistics.

Huck, M., Riess, S., and Fraser, A. (2017). Target-side word segmentation strategies for neural machine translation. In *Proceedings of the Second Conference on Machine Translation*, pages 56–67, Copenhagen, Denmark. Association for Computational Linguistics.

Iglesias, G., Allauzen, C., Byrne, B., de Gispert, A., and Riley, M. D. (2011). Hierarchical phrase-based translation representations. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1373–1383, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Iglesias, G., de Gispert, A., Banga, E. R., and Byrne, B. (2009). Hierarchical phrase-based translation with weighted finite state transducers. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 433–441, Boulder, Colorado. Association for Computational Linguistics.

Iglesias, G., de Gispert, A., and Byrne, B. (2015). Transducer disambiguation with sparse topological features. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2275–2280, Lisbon, Portugal. Association for Computational Linguistics.

Iglesias, G., Tambellini, W., de Gispert, A., Hasler, E., and Byrne, B. (2018). Accelerating NMT batched beam decoding with LMBR posteriors for deployment. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers)*, pages 106–113, New Orleans - Louisiana. Association for Computational Linguistics.

Im, J. and Cho, S. (2017). Distance-based self-attention network for natural language inference. *arXiv preprint arXiv:1712.02047*.

Imamura, K., Fujita, A., and Sumita, E. (2018). Enhancement of encoder and attention using target monolingual corpora in neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 55–63, Melbourne, Australia. Association for Computational Linguistics.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 448–456. JMLR.org.

Isabelle, P., Cherry, C., and Foster, G. (2017). A challenge set approach to evaluating machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2486–2496, Copenhagen, Denmark. Association for Computational Linguistics.

Ishiwatari, S., Yao, J., Liu, S., Li, M., Zhou, M., Yoshinaga, N., Kitsuregawa, M., and Jia, W. (2017). Chunk-based decoder for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1901–1912, Vancouver, Canada. Association for Computational Linguistics.

Jain, S. and Wallace, B. C. (2019). Attention is not Explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3543–3556, Minneapolis, Minnesota. Association for Computational Linguistics.

Jauregi Unanue, I., Garmendia Arratibel, L., Zare Borzeshi, E., and Piccardi, M. (2018). English-Basque statistical and neural machine translation. In *Proceedings of the 11th Language Resources and Evaluation Conference*, Miyazaki, Japan. European Language Resource Association.

Jean, S., Cho, K., Memisevic, R., and Bengio, Y. (2015a). On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10, Beijing, China. Association for Computational Linguistics.

Jean, S., Firat, O., Cho, K., Memisevic, R., and Bengio, Y. (2015b). Montreal neural machine translation systems for WMT'15. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 134–140, Lisbon, Portugal. Association for Computational Linguistics.

Jean, S., Lauly, S., Firat, O., and Cho, K. (2017). Does neural machine translation benefit from larger context? *arXiv preprint arXiv:1704.05135*.

Ji, J., Wang, Q., Toutanova, K., Gong, Y., Truong, S., and Gao, J. (2017). A nested attention neural hybrid model for grammatical error correction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 753–762. Association for Computational Linguistics.

Jia, Y., Carl, M., and Wang, X. (2019a). Post-editing neural machine translation versus phrase-based machine translation for English–Chinese. *Machine Translation*, pages 1–21.

Jia, Y., Weiss, R. J., et al. (2019b). Introducing Translatotron: An end-to-end speech-to-speech translation model. https://ai.googleblog.com/2019/05/introducing-translatotron-end-to-end.html.

Johansen, A. R., Hansen, J. M., Obeid, E. K., Sønderby, C. K., and Winther, O. (2016). Neural machine translation with characters and hierarchical encoding. *arXiv preprint arXiv:1610.06550*.

Johnson, M., Schuster, M., Le, Q. V., Krikun, M., Wu, Y., Chen, Z., Thorat, N., Viégas, F., Wattenberg, M., Corrado, G., Hughes, M., and Dean, J. (2017). Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Joulin, A. and Mikolov, T. (2015). Inferring algorithmic patterns with stack-augmented recurrent nets. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28*, pages 190–198. Curran Associates, Inc.

Junczys-Dowmunt, M. (2018a). Dual conditional cross-entropy filtering of noisy parallel corpora. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 888–895, Belgium, Brussels. Association for Computational Linguistics.

Junczys-Dowmunt, M. (2018b). Microsoft's submission to the WMT2018 news translation task: How I learned to stop worrying and love the data. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 425–430, Belgium, Brussels. Association for Computational Linguistics.

Junczys-Dowmunt, M., Dwojak, T., and Hoang, H. (2016a). Is neural machine translation ready for deployment? A case study on 30 translation directions. In *International Workshop on Spoken Language Translation IWSLT*.

Junczys-Dowmunt, M., Dwojak, T., and Sennrich, R. (2016b). The AMU-UEDIN submission to the WMT16 news translation task: Attention-based NMT models as feature functions in phrase-based SMT. In *Proceedings of the First Conference on Machine Translation*, pages 319–325, Berlin, Germany. Association for Computational Linguistics.

Junczys-Dowmunt, M. and Grundkiewicz, R. (2016). Phrase-based machine translation is state-of-the-art for automatic grammatical error correction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1546–1556. Association for Computational Linguistics.

Kaiser, Ł. and Bengio, S. (2016). Can active memory replace attention? In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems 29*, pages 3781–3789. Curran Associates, Inc.

Kaiser, Ł., Gomez, A. N., and Chollet, F. (2017). Depthwise separable convolutions for neural machine translation. *arXiv preprint arXiv:1706.03059*.

Kaiser, Ł., Roy, A., Vaswani, A., Pamar, N., Bengio, S., Uszkoreit, J., and Shazeer, N. (2018). Fast decoding in sequence models using discrete latent variables. *arXiv preprint arXiv:1803.03382*.

Kalchbrenner, N. and Blunsom, P. (2013). Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, Seattle, Washington, USA. Association for Computational Linguistics.

Kalchbrenner, N., Espeholt, L., Simonyan, K., van den Oord, A., Graves, A., and Kavukcuoglu, K. (2016). Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099*.

Kalchbrenner, N., Grefenstette, E., and Blunsom, P. (2014). A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 655–665. Association for Computational Linguistics.

Kann, K., Cotterell, R., and Schütze, H. (2017). Neural multi-source morphological reinflection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 514–524, Valencia, Spain. Association for Computational Linguistics.

Karpathy, A., Johnson, J., and Fei-Fei, L. (2015). Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*.

Karpukhin, V., Levy, O., Eisenstein, J., and Ghazvininejad, M. (2019). Training on synthetic noise improves robustness to natural noise in machine translation. *arXiv preprint arXiv:1902.01509*.

Kazimi, M. B. and Costa-Jussá, M. R. (2017). Coverage for character based neural machine translation. *Procesamiento del Lenguaje Natural*, 59(0):99–106.

Kell, G. (2018). Overcoming catastrophic forgetting in neural machine translation. MPhil dissertation, University of Cambridge.

Keneshloo, Y., Shi, T., Reddy, C. K., and Ramakrishnan, N. (2018). Deep reinforcement learning for sequence to sequence models. *arXiv preprint arXiv:1805.09461*.

Khayrallah, H. and Koehn, P. (2018). On the impact of various types of noise on neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 74–83, Melbourne, Australia. Association for Computational Linguistics.

Khayrallah, H., Kumar, G., Duh, K., Post, M., and Koehn, P. (2017). Neural lattice search for domain adaptation in machine translation. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 20–25, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Khayrallah, H., Thompson, B., Duh, K., and Koehn, P. (2018). Regularized training objective for continued training for domain adaptation in neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 36–44, Melbourne, Australia. Association for Computational Linguistics.

Kiefer, J. (1953). Sequential minimax search for a maximum. *Proceedings of the American mathematical society*, 4(3):502–506.

Kikuchi, Y., Neubig, G., Sasano, R., Takamura, H., and Okumura, M. (2016). Controlling output length in neural encoder-decoders. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1328–1338, Austin, Texas. Association for Computational Linguistics.

Kim, H.-G., Na, H., Lee, H., Lee, J., Kang, T. G., Lee, M.-J., and Choi, Y. S. (2019a). Knowledge distillation using output errors for self-attention end-to-end models. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6181–6185.

Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. Association for Computational Linguistics.

Kim, Y., Jernite, Y., Sontag, D., and Rush, A. M. (2016). Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 2741–2749. AAAI Press.

Kim, Y. and Rush, A. M. (2016). Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327, Austin, Texas. Association for Computational Linguistics.

Kim, Y., Rush, A. M., Yu, L., Kuncoro, A., Dyer, C., and Melis, G. (2019b). Unsupervised recurrent neural network grammars. *arXiv preprint arXiv:1904.03746*.

Kingma, D. P. and Ba, J. L. (2015). Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*. Amsterdam Machine Learning lab (IVI, FNWI).

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526.

Klein, G., Kim, Y., Deng, Y., Senellart, J., and Rush, A. M. (2017). OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada. Association for Computational Linguistics.

Knight, K. (1999). A statistical MT tutorial workbook.

Kobus, C., Crego, J., and Senellart, J. (2017). Domain control for neural machine translation. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pages 372–378, Varna, Bulgaria. INCOMA Ltd.

Koehn, P. (2010). *Statistical Machine Translation*. Cambridge University Press, New York, NY, USA, 1st edition.

Koehn, P. (2017). Neural machine translation. *arXiv preprint arXiv:1709.07809*.

Koehn, P. and Hoang, H. (2007). Factored translation models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 868–876, Prague, Czech Republic. Association for Computational Linguistics.

Koehn, P., Khayrallah, H., Heafield, K., and Forcada, M. L. (2018). Findings of the WMT 2018 shared task on parallel corpus filtering. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 726–739, Belgium, Brussels. Association for Computational Linguistics.

Koehn, P. and Knowles, R. (2017). Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39, Vancouver. Association for Computational Linguistics.

Koncel-Kedziorski, R., Bekal, D., Luan, Y., Lapata, M., and Hajishirzi, H. (2019). Text Generation from Knowledge Graphs with Graph Transformers. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2284–2293, Minneapolis, Minnesota. Association for Computational Linguistics.

Kong, X., Tu, Z., Shi, S., Hovy, E., and Zhang, T. (2018). Neural machine translation with adequacy-oriented learning. *arXiv preprint arXiv:1811.08541*.

Krause, B., Kahembwe, E., Murray, I., and Renals, S. (2019). Dynamic evaluation of Transformer language models. *arXiv preprint arXiv:1904.08378*.

Kreutzer, J. and Sokolov, A. (2018). Optimally segmenting inputs for NMT shows preference for character-level processing. *arXiv preprint arXiv:1810.01480*.

Kuang, S., Xiong, D., Luo, W., and Zhou, G. (2017). Cache-based document-level neural machine translation. *arXiv preprint arXiv:1711.11221*.

Kuchaiev, O., Ginsburg, B., Gitman, I., Lavrukhin, V., Case, C., and Micikevicius, P. (2018). OpenSeq2Seq: Extensible toolkit for distributed and mixed precision training of sequence-to-sequence models. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 41–46, Melbourne, Australia. Association for Computational Linguistics.

Kudo, T. (2018). Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics.

Kudo, T. and Richardson, J. (2018). SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Kumar, A. and Sarawagi, S. (2019). Calibration of encoder decoder models for neural machine translation. *arXiv preprint arXiv:1903.00802*.

Kumar, G., Foster, G., Cherry, C., and Krikun, M. (2019). Reinforcement learning based curriculum optimization for neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2054–2061, Minneapolis, Minnesota. Association for Computational Linguistics.

Kumar, S. and Byrne, B. (2004). Minimum Bayes-risk decoding for statistical machine translation. In *HLT-NAACL 2004: Main Proceedings*, pages 169–176, Boston, Massachusetts, USA. Association for Computational Linguistics.

Kumar, S. and Byrne, B. (2005). Local phrase reordering models for statistical machine translation. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 161–168, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Kunchukuttan, A. and Bhattacharyya, P. (2016). Faster decoding for subword level phrase-based SMT between related languages. In *Proceedings of the Third Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial3)*, pages 82–88, Osaka, Japan. The COLING 2016 Organizing Committee.

Kunchukuttan, A. and Bhattacharyya, P. (2017). Learning variable length units for SMT between related languages via byte pair encoding. In *Proceedings of the First Workshop on Subword and Character Level Models in NLP*, pages 14–24, Copenhagen, Denmark. Association for Computational Linguistics.

Kurach, K., Andrychowicz, M., and Sutskever, I. (2015). Neural random-access machines. *arXiv preprint arXiv:1511.06392*.

Lakew, S. M., Cettolo, M., and Federico, M. (2018). A comparison of Transformer and recurrent neural networks on multilingual neural machine translation. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 641–652, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Lamb, A. M., Goyal, A. G. A. P., Zhang, Y., Zhang, S., Courville, A. C., and Bengio, Y. (2016). Professor forcing: A new algorithm for training recurrent networks. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems 29*, pages 4601–4609. Curran Associates, Inc.

Lample, G., Conneau, A., Denoyer, L., and Ranzato, M. (2017). Unsupervised machine translation using monolingual corpora only. *arXiv preprint arXiv:1711.00043*.

Lample, G., Ott, M., Conneau, A., Denoyer, L., and Ranzato, M. (2018). Phrase-based & neural unsupervised machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5039–5049, Brussels, Belgium. Association for Computational Linguistics.

Larochelle, H. and Hinton, G. E. (2010). Learning to combine foveal glimpses with a third-order Boltzmann machine. In *Advances in neural information processing systems*, pages 1243–1251.

Läubli, S., Sennrich, R., and Volk, M. (2018). Has machine translation achieved human parity? a case for document-level evaluation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4791–4796, Brussels, Belgium. Association for Computational Linguistics.

Lawson, D., Chiu, C.-C., Tucker, G., Raffel, C., Swersky, K., and Jaitly, N. (2018). Learning hard alignments with variational inference. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5799–5803.

Le, H.-S., Allauzen, A., and Yvon, F. (2012). Continuous space translation models with neural networks. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 39–48, Montréal, Canada. Association for Computational Linguistics.

Le, Q. V., Luong, M.-T., Sutskever, I., Vinyals, O., and Zaremba, W. (2016). Neural machine translation systems with rare word processing. US Patent App. 14/921,925.

Lecorvé, G. and Motlicek, P. (2012). Conversion of recurrent neural network language models to weighted finite state transducers for automatic speech recognition. In *Thirteenth Annual Conference of the International Speech Communication Association*.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. E., and Jackel, L. D. (1989a). Backpropagation applied to handwritten zip code recognition. *Neural Comput.*, 1(4):541–551.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. E., and Jackel, L. D. (1990). Handwritten digit recognition with a back-propagation network. In Touretzky, D. S., editor, *Advances in Neural Information Processing Systems 2*, pages 396–404. Morgan-Kaufmann.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

LeCun, Y., Denker, J. S., Solla, S. A., Howard, R. E., and Jackel, L. D. (1989b). Optimal brain damage. In *Advances in neural information processing systems*, volume 2, pages 598–605.

Lee, C.-Y. and Osindero, S. (2016). Recursive recurrent nets with attention modeling for OCR in the wild. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Lee, J., Cho, K., and Hofmann, T. (2017). Fully character-level neural machine translation without explicit segmentation. *Transactions of the Association for Computational Linguistics*, 5:365–378.

Lee, J., Mansimov, E., and Cho, K. (2018). Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1173–1182, Brussels, Belgium. Association for Computational Linguistics.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions and reversals. In *Soviet physics doklady*, volume 10, page 707.

Levin, P., Dhanuka, N., Khalil, T., Kovalev, F., and Khalilov, M. (2017). Toward a full-scale neural machine translation in production: the booking.com use case. *arXiv preprint arXiv:1709.05820*.

Lewis, W. D. (2015). Skype translator: Breaking down language and hearing barriers. *Translating and the Computer (TC37)*, 10:125–149.

Lewis II, P. M. and Stearns, R. E. (1968). Syntax-directed transduction. *J. ACM*, 15(3):465–488.

L'Hostis, G., Grangier, D., and Auli, M. (2016). Vocabulary selection strategies for neural machine translation. *arXiv preprint arXiv:1610.00072*.

Li, A., Zhang, S., Wang, D., and Zheng, T. F. (2017a). Enhanced neural machine translation by learning from draft. In *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 1583–1587.

Li, F., Quan, D., Qiang, W., Tong, X., and Zhu, J. (2017b). Handling many-to-one unk translation for neural machine translation. In *Machine Translation: 13th China Workshop, CWMT 2017, Revised Selected Papers*, pages 102–111. Springer.

Li, H. and Chen, H. (2019). Human vs. AI: An assessment of the translation quality between translators and machine translation. *International Journal of Translation, Interpretation, and Applied Linguistics (IJTIAL)*, 1(1):43–54.

Li, J., Chen, X., Hovy, E., and Jurafsky, D. (2016a). Visualizing and understanding neural models in NLP. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 681–691, San Diego, California. Association for Computational Linguistics.

Li, J., Galley, M., Brockett, C., Gao, J., and Dolan, B. (2016b). A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119, San Diego, California. Association for Computational Linguistics.

Li, J. and Jurafsky, D. (2016). Mutual information and diverse decoding improve neural machine translation. *arXiv preprint arXiv:1601.00372*.

Li, J., Monroe, W., Shi, T., Jean, S., Ritter, A., and Jurafsky, D. (2017c). Adversarial learning for neural dialogue generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2157–2169, Copenhagen, Denmark. Association for Computational Linguistics.

Li, P., Liu, Y., and Sun, M. (2013). Recursive autoencoders for ITG-based translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 567–577, Seattle, Washington, USA. Association for Computational Linguistics.

Li, S., Xu, J., Zhang, Y., and Chen, Y. (2017d). A method of unknown words processing for neural machine translation using HowNet. In *Machine Translation: 13th China Workshop, CWMT 2017, Revised Selected Papers*, pages 20–29. Springer.

Li, X., Zhang, J., and Zong, C. (2016c). Towards zero unknown word in neural machine translation. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 2852–2858. AAAI Press.

Li, Y., Liu, X., Liu, D., Zhang, X., and Liu, J. (2019). Learning efficient lexically-constrained neural machine translation with external memory. *arXiv preprint arXiv:1901.11344*.

Li, Y., Xiao, T., Li, Y., Wang, Q., Xu, C., and Zhu, J. (2018). A simple and effective approach to coverage-aware neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 292–297, Melbourne, Australia. Association for Computational Linguistics.

Li, Y., Xiong, D., and Zhang, M. (2017e). Neural machine translation with phrasal attention. In *Machine Translation: 13th China Workshop, CWMT 2017, Revised Selected Papers*, pages 1–8. Springer.

Libovický, J. and Helcl, J. (2018). End-to-end non-autoregressive neural machine translation with connectionist temporal classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3016–3021, Brussels, Belgium. Association for Computational Linguistics.

Lin, J. and Dyer, C. (2010). Data-intensive text processing with MapReduce. In *NAACL HLT 2010 Tutorial Abstracts*, pages 1–2, Los Angeles, California. Association for Computational Linguistics.

Lin, J., Sun, X., Ren, X., Li, M., and Su, Q. (2018a). Learning when to concentrate or divert attention: Self-adaptive attention temperature for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2985–2990, Brussels, Belgium. Association for Computational Linguistics.

Lin, J., Sun, X., Ren, X., Ma, S., Su, J., and Su, Q. (2018b). Deconvolution-based global decoding for neural machine translation. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3260–3271, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Lin, Z., Feng, M., Santos, C. N. d., Yu, M., Xiang, B., Zhou, B., and Bengio, Y. (2017). A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Ling, W., Trancoso, I., Dyer, C., and Black, A. W. (2015). Character-based neural machine translation. *arXiv preprint arXiv:1511.04586*.

Lipton, Z. C. (2018). The mythos of model interpretability. *Queue*, 16(3):30:31–30:57.

Liu, L., Utiyama, M., Finch, A., and Sumita, E. (2016a). Agreement on target-bidirectional neural machine translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 411–416, San Diego, California. Association for Computational Linguistics.

Liu, L., Utiyama, M., Finch, A., and Sumita, E. (2016b). Neural machine translation with supervised attention. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3093–3102, Osaka, Japan. The COLING 2016 Organizing Committee.

Liu, N. F., May, J., Pust, M., and Knight, K. (2018a). Augmenting statistical machine translation with subword translation of out-of-vocabulary words. *arXiv preprint arXiv:1808.05700*.

Liu, X., Wang, Y., Chen, X., Gales, M. J., and Woodland, P. C. (2014). Efficient lattice rescoring using recurrent neural network language models. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4908–4912.

Liu, Y., Luo, Z., and Zhu, K. (2018b). Controlling length in abstractive summarization using a convolutional neural network. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4110–4119, Brussels, Belgium. Association for Computational Linguistics.

Liu, Y., Mi, H., Feng, Y., and Liu, Q. (2009). Joint decoding with multiple translation models. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 576–584, Suntec, Singapore. Association for Computational Linguistics.

Liu, Y., Sun, C., Lin, L., and Wang, X. (2016c). Learning natural language inference using bidirectional LSTM model and inner-attention. *arXiv preprint arXiv:1605.09090*.

Liu, Y., Xiong, H., He, Z., Zhang, J., Wu, H., Wang, H., and Zong, C. (2019). End-to-end speech translation with knowledge distillation. *arXiv preprint arXiv:1904.08075*.

Liu, Y., Zhang, Y., Che, W., and Qin, B. (2015). Transition-based syntactic linearization. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 113–122, Denver, Colorado. Association for Computational Linguistics.

Liu, Y., Zhou, L., Wang, Y., Zhao, Y., Zhang, J., and Zong, C. (2018c). A comparable study on model averaging, ensembling and reranking in NMT. In Zhang, M., Ng, V., Zhao, D., Li, S., and Zan, H., editors, *Natural Language Processing and Chinese Computing*, pages 299–308, Cham. Springer International Publishing.

Livni, R., Shalev-Shwartz, S., and Shamir, O. (2014). On the computational efficiency of training neural networks. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 855–863. Curran Associates, Inc.

Long, Z., Utsuro, T., Mitsuhashi, T., and Yamamoto, M. (2016). Translation of patent sentences with a large vocabulary of technical terms using neural machine translation. In *Proceedings of the 3rd Workshop on Asian Translation (WAT2016)*, pages 47–57, Osaka, Japan. The COLING 2016 Organizing Committee.

Lu, Y., Keung, P., Ladhak, F., Bhardwaj, V., Zhang, S., and Sun, J. (2018). A neural interlingua for multilingual machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 84–92, Belgium, Brussels. Association for Computational Linguistics.

Lu, Z., Sindhwani, V., and Sainath, T. N. (2016). Learning compact recurrent neural networks. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5960–5964.

Luong, M.-T., Brevdo, E., and Zhao, R. (2017). Neural machine translation (seq2seq) tutorial. *https://github.com/tensorflow/nmt*.

Luong, M.-T., Le, Q. V., Sutskever, I., Vinyals, O., and Kaiser, Ł. (2015a). Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*.

Luong, M.-T. and Manning, C. D. (2015). Stanford neural machine translation systems for spoken language domains. In *Proceedings of the International Workshop on Spoken Language Translation*, pages 76–79.

Luong, M.-T. and Manning, C. D. (2016). Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1054–1063, Berlin, Germany. Association for Computational Linguistics.

Luong, M.-T., Pham, H., and Manning, C. D. (2015b). Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421. Association for Computational Linguistics.

Luong, M.-T., Sutskever, I., Le, Q. V., Vinyals, O., and Zaremba, W. (2015c). Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 11–19, Beijing, China. Association for Computational Linguistics.

Ma, C., Liu, L., Tamura, A., Zhao, T., and Sumita, E. (2017). Deterministic attention for sequence-to-sequence constituent parsing. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, pages 3237–3243. AAAI Press.

Ma, C., Tamura, A., Utiyama, M., Zhao, T., and Sumita, E. (2018a). Forest-based neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1253–1263, Melbourne, Australia. Association for Computational Linguistics.

Ma, M., Huang, L., Xiong, H., Liu, K., Zhang, C., He, Z., Liu, H., Li, X., and Wang, H. (2018b). Stacl: Simultaneous translation with integrated anticipation and controllable latency. *arXiv preprint arXiv:1810.08398*.

Ma, X., Li, K., and Koehn, P. (2018c). An analysis of source context dependency in neural machine translation. In *21st Annual Conference of the European Association for Machine Translation*, page 189.

Macháček, D., Vidra, J., and Bojar, O. (2018). Morphological and language-agnostic word segmentation for NMT. In Sojka, P., Horák, A., Kopeček, I., and Pala, K., editors, *Text, Speech, and Dialogue*, pages 277–284, Cham. Springer International Publishing.

Macherey, W., Och, F. J., Thayer, I., and Uszkoreit, J. (2008). Lattice-based minimum error rate training for statistical machine translation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 725–734, Honolulu, Hawaii. Association for Computational Linguistics.

Macketanz, V., Avramidis, E., Burchardt, A., and Uszkoreit, H. (2018). Fine-grained evaluation of German-English machine translation based on a test suite. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 578–587, Belgium, Brussels. Association for Computational Linguistics.

Mahata, S. K., Mandal, S., Das, D., and Bandyopadhyay, S. (2018). SMT vs NMT: a comparison over Hindi & Bengali simple sentences. *arXiv preprint arXiv:1812.04898*.

Maillard, J., Clark, S., and Yogatama, D. (2017). Jointly learning sentence embeddings and syntax with unsupervised Tree-LSTMs. *arXiv preprint arXiv:1705.09189*.

Malaviya, C., Ferreira, P., and Martins, A. F. T. (2018). Sparse and constrained attention for neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 370–376, Melbourne, Australia. Association for Computational Linguistics.

Marcheggiani, D., Bastings, J., and Titov, I. (2018). Exploiting semantics in neural machine translation with graph convolutional networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 486–492, New Orleans, Louisiana. Association for Computational Linguistics.

Marcus, M. P., Marcinkiewicz, M. A., and Santorini, B. (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics*, 19(2):313–330.

Marie, B. and Fujita, A. (2018). A smorgasbord of features to combine phrase-based and neural machine translation. In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Papers)*, pages 111–124, Boston, MA. Association for Machine Translation in the Americas.

Maruf, S. and Haffari, G. (2018). Document context neural machine translation with memory networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1275–1284, Melbourne, Australia. Association for Computational Linguistics.

Maruf, S., Martins, A. F. T., and Haffari, G. (2019). Selective attention for context-aware neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3092–3102, Minneapolis, Minnesota. Association for Computational Linguistics.

McCandlish, S., Kaplan, J., Amodei, D., and Team, O. D. (2018). An empirical model of large-batch training. *arXiv preprint arXiv:1812.06162*.

McCann, B., Bradbury, J., Xiong, C., and Socher, R. (2017). Learned in translation: Contextualized word vectors. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 6294–6305. Curran Associates, Inc.

Medina, J. R. and Kalita, J. (2018). Parallel attention mechanisms in neural machine translation. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 547–552.

Mehri, S. and Sigal, L. (2018). Middle-out decoding. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31*, pages 5518–5529. Curran Associates, Inc.

Melamed, I. D. (2003). Multitext grammars and synchronous parsers. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*.

Melamed, I. D., Satta, G., and Wellington, B. (2004). Generalized multitext grammars. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*.

Menacer, M. A., Langlois, D., Mella, O., Fohr, D., Jouvet, D., and Smaïli, K. (2017). Is statistical machine translation approach dead? In *ICNLSSP 2017 - International Conference on Natural Language, Signal and Speech Processing*, pages 1–5, Casablanca, Morocco. ISGA.

Meng, F., Tu, Z., Cheng, Y., Wu, H., Zhai, J., Yang, Y., and Wang, D. (2018). Neural machine translation with key-value memory-augmented attention. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, pages 2574–2580. AAAI Press.

Mi, H., Sankaran, B., Wang, Z., and Ittycheriah, A. (2016a). Coverage embedding models for neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 955–960, Austin, Texas. Association for Computational Linguistics.

Mi, H., Wang, Z., and Ittycheriah, A. (2016b). Supervised attentions for neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2283–2288, Austin, Texas. Association for Computational Linguistics.

Mi, H., Wang, Z., and Ittycheriah, A. (2016c). Vocabulary manipulation for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 124–129, Berlin, Germany. Association for Computational Linguistics.

Miao, G., Xu, J., Li, Y., Li, S., and Chen, Y. (2017). An unknown word processing method in NMT by integrating syntactic structure and semantic concept. In *Machine Translation: 13th China Workshop, CWMT 2017, Revised Selected Papers*, pages 43–54. Springer.

Miceli Barone, A. V., Haddow, B., Germann, U., and Sennrich, R. (2017). Regularization techniques for fine-tuning in neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1489–1494, Copenhagen, Denmark. Association for Computational Linguistics.

Michel, P., Li, X., Neubig, G., and Pino, J. (2019). On evaluation of adversarial perturbations for sequence-to-sequence models. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3103–3114, Minneapolis, Minnesota. Association for Computational Linguistics.

Michel, P. and Neubig, G. (2018). MTNT: A testbed for machine translation of noisy text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 543–553, Brussels, Belgium. Association for Computational Linguistics.

Miculicich, L., Pappas, N., Ram, D., and Popescu-Belis, A. (2018a). Self-attentive residual decoder for neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1366–1379, New Orleans, Louisiana. Association for Computational Linguistics.

Miculicich, L., Ram, D., Pappas, N., and Henderson, J. (2018b). Document-level neural machine translation with hierarchical attention networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2947–2954, Brussels, Belgium. Association for Computational Linguistics.

Mieno, T., Neubig, G., Sakti, S., Toda, T., and Nakamura, S. (2015). Speed or accuracy? A study in evaluation of simultaneous speech translation. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Mikolov, T., Karafiát, M., Burget, L., Černockỳ, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*.

Mikolov, T., Le, Q. V., and Sutskever, I. (2013b). Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

Mino, H., Utiyama, M., Sumita, E., and Tokunaga, T. (2017). Key-value attention mechanism for neural machine translation. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 290–295, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Mizumoto, T., Hayashibe, Y., Komachi, M., Nagata, M., and Matsumoto, Y. (2012). The effect of learner corpus size in grammatical error correction of ESL writings. In *Proceedings of COLING 2012: Posters*, pages 863–872. The COLING 2012 Organizing Committee.

Mnih, A. and Kavukcuoglu, K. (2013). Learning word embeddings efficiently with noise-contrastive estimation. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 26*, pages 2265–2273. Curran Associates, Inc.

Mnih, A. and Teh, Y. W. (2012). A fast and simple algorithm for training neural probabilistic language models. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, pages 419–426, USA. Omnipress.

Mnih, V., Heess, N., Graves, A., et al. (2014). Recurrent models of visual attention. In *Advances in Neural Information Processing Systems*, pages 2204–2212.

Mohri, M. (1997). Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311.

Mohri, M. (2003). Edit-distance of weighted automata: General definitions and algorithms. *International Journal of Foundations of Computer Science*, 14(06):957–982.

Mohri, M., Pereira, F., and Riley, M. D. (2008). *Speech Recognition with Weighted Finite-State Transducers*, pages 559–584. Springer Berlin Heidelberg, Berlin, Heidelberg.

Mohri, M. and Riley, M. D. (2001). A weight pushing algorithm for large vocabulary speech recognition. In *Interspeech*, pages 1603–1606.

Mohri, M. and Riley, M. D. (2015). On the disambiguation of weighted automata. In Drewes, F., editor, *Implementation and Application of Automata*, pages 263–278, Cham. Springer International Publishing.

Montavon, G., Samek, W., and Müller, K.-R. (2018). Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 73:1–15.

Morishita, M., Oda, Y., Neubig, G., Yoshino, K., Sudoh, K., and Nakamura, S. (2017). An empirical study of mini-batch creation strategies for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 61–68, Vancouver. Association for Computational Linguistics.

Mou, L., Men, R., Li, G., Xu, Y., Zhang, L., Yan, R., and Jin, Z. (2016). Natural language inference by tree-based convolution and heuristic matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 130–136. Association for Computational Linguistics.

Moussallem, D., Arčan, M., Ngomo, A.-C. N., and Buitelaar, P. (2019). Augmenting neural machine translation with knowledge graphs. *arXiv preprint arXiv:1902.08816*.

Müller, M., Nguyen, T. S., Niehues, J., Cho, E., Krüger, B., Ha, T.-L., Kilgour, K., Sperber, M., Mediani, M., Stüker, S., and Waibel, A. (2016). Lecture translator - speech translation framework for simultaneous lecture translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 82–86, San Diego, California. Association for Computational Linguistics.

Müller, M., Rios, A., Voita, E., and Sennrich, R. (2018). A large-scale test set for the evaluation of context-aware pronoun translation in neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 61–72, Belgium, Brussels. Association for Computational Linguistics.

Murray, K. and Chiang, D. (2018). Correcting length bias in neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 212–223, Belgium, Brussels. Association for Computational Linguistics.

Murthy, R., Kunchukuttan, A., and Bhattacharyya, P. (2019). Addressing word-order divergence in multilingual neural machine translation for extremely low resource languages. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3868–3873, Minneapolis, Minnesota. Association for Computational Linguistics.

Nadejde, M., Reddy, S., Sennrich, R., Dwojak, T., Junczys-Dowmunt, M., Koehn, P., and Birch, A. (2017). Predicting target language CCG supertags improves neural machine translation. In *Proceedings of the Second Conference on Machine Translation*, pages 68–79, Copenhagen, Denmark. Association for Computational Linguistics.

Nakazawa, T., Yaguchi, M., Uchimoto, K., Utiyama, M., Sumita, E., Kurohashi, S., and Isahara, H. (2016). ASPEC: Asian scientific paper excerpt corpus. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 2204–2208, Portorož, Slovenia. European Language Resources Association (ELRA).

Napoles, C., Sakaguchi, K., Post, M., and Tetreault, J. (2015). Ground truth for grammatical error correction metrics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 588–593. Association for Computational Linguistics.

Napoles, C., Sakaguchi, K., and Tetreault, J. (2017). JFLEG: A fluency corpus and benchmark for grammatical error correction. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 229–234, Valencia, Spain. Association for Computational Linguistics.

Neishi, M., Sakuma, J., Tohda, S., Ishiwatari, S., Yoshinaga, N., and Toyoda, M. (2017). A bag of useful tricks for practical neural machine translation: Embedding layer initialization and large batch size. In *Proceedings of the 4th Workshop on Asian Translation (WAT2017)*, pages 99–109, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Neubig, G. (2011). The Kyoto free translation task. http://www.phontron.com/kftt.

Neubig, G. (2013). Travatar: A forest-to-string machine translation engine based on tree transducers. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 91–96, Sofia, Bulgaria. Association for Computational Linguistics.

Neubig, G. (2014). Forest-to-string SMT for Asian language translation: NAIST at WAT 2014. In *Proceedings of the 1st Workshop on Asian Translation (WAT2014)*, pages 20–25, Tokyo, Japan. Workshop on Asian Translation.

Neubig, G. (2016). Lexicons and minimum risk training for neural machine translation: NAIST-CMU at WAT2016. In *Proceedings of the 3rd Workshop on Asian Translation (WAT2016)*, pages 119–125, Osaka, Japan. The COLING 2016 Organizing Committee.

Neubig, G. (2017). Neural machine translation and sequence-to-sequence models: A tutorial. *arXiv preprint arXiv:1703.01619*.

Neubig, G. and Duh, K. (2014). On the elements of an accurate tree-to-string machine translation system. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 143–149, Baltimore, Maryland. Association for Computational Linguistics.

Neubig, G. and Hu, J. (2018). Rapid adaptation of neural machine translation to new languages. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 875–880, Brussels, Belgium. Association for Computational Linguistics.

Neubig, G., Morishita, M., and Nakamura, S. (2015). Neural reranking improves subjective quality of machine translation: NAIST at WAT2015. In *Proceedings of the 2nd Workshop on Asian Translation (WAT2015)*, pages 35–41, Kyoto, Japan. Workshop on Asian Translation.

Neubig, G., Sperber, M., Wang, X., Felix, M., Matthews, A., Padmanabhan, S., Qi, Y., Sachan, D., Arthur, P., Godard, P., Hewitt, J., Riad, R., and Wang, L. (2018). XNMT: The eXtensible neural machine translation toolkit. In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Papers)*, pages 185–192, Boston, MA. Association for Machine Translation in the Americas.

Neves, M., Yepes, A. J., and Névéol, A. (2016). The scielo corpus: A parallel corpus of scientific publications for biomedicine. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 2942–2948, Portorož, Slovenia. European Language Resources Association (ELRA).

Ng, H. T., Wu, S. M., Briscoe, T., Hadiwinoto, C., Susanto, R. H., and Bryant, C. (2014). The CoNLL-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–14, Baltimore, Maryland. Association for Computational Linguistics.

Ng, H. T., Wu, S. M., Wu, Y., Hadiwinoto, C., and Tetreault, J. (2013). The CoNLL-2013 shared task on grammatical error correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–12, Sofia, Bulgaria. Association for Computational Linguistics.

Nguyen, T. Q. and Chiang, D. (2017). Transfer learning across low-resource, related languages for neural machine translation. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 296–301, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Nguyen, T. Q. and Chiang, D. (2018). Improving lexical choice in neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 334–343, New Orleans, Louisiana. Association for Computational Linguistics.

Niehues, J., Cho, E., Ha, T.-L., and Waibel, A. (2016). Pre-translation for neural machine translation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1828–1836, Osaka, Japan. The COLING 2016 Organizing Committee.

Niehues, J., Cho, E., Ha, T.-L., and Waibel, A. (2017). Analyzing neural MT search and model performance. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 11–17, Vancouver. Association for Computational Linguistics.

Nishimura, Y., Sudoh, K., Neubig, G., and Nakamura, S. (2018). Multi-source neural machine translation with data augmentation. *arXiv preprint arXiv:1810.06826*.

Niu, X., Denkowski, M., and Carpuat, M. (2018). Bi-Directional neural machine translation with synthetic parallel data. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 84–91, Melbourne, Australia. Association for Computational Linguistics.

Och, F. J. (2003). Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, Sapporo, Japan. Association for Computational Linguistics.

Och, F. J. and Ney, H. (2001). Statistical multi-source translation. In *Proceedings of MT Summit*, volume 8, pages 253–258.

Och, F. J. and Ney, H. (2002). Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 295–302, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Och, F. J. and Ney, H. (2003). A systematic comparison of various statistical alignment models. *American Journal of Computational Linguistics*, 29(1):19–51.

Oda, Y., Neubig, G., Sakti, S., Toda, T., and Nakamura, S. (2015). Ckylark: A more robust PCFG-LA parser. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 41–45, Denver, Colorado. Association for Computational Linguistics.

Ojha, A. K., Chowdhury, K. D., Liu, C.-H., and Saxena, K. (2018). The RGNLP machine translation systems for WAT 2018. *arXiv preprint arXiv:1812.00798*.

Östling, R. and Tiedemann, J. (2017). Neural machine translation for low-resource languages. *arXiv preprint arXiv:1708.05729*.

Ott, M., Auli, M., Grangier, D., and Ranzato, M. (2018a). Analyzing uncertainty in neural machine translation. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3956–3965, Stockholmsmässan, Stockholm Sweden. PMLR.

Ott, M., Edunov, S., Baevski, A., Fan, A., Gross, S., Ng, N., Grangier, D., and Auli, M. (2019). fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*.

Ott, M., Edunov, S., Grangier, D., and Auli, M. (2018b). Scaling neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 1–9, Belgium, Brussels. Association for Computational Linguistics.

Pal, S., Zampieri, M., and van Genabith, J. (2016). USAAR: An operation sequential model for automatic statistical post-editing. In *Proceedings of the First Conference on Machine Translation*, pages 759–763, Berlin, Germany. Association for Computational Linguistics.

Palm, R., Paquet, U., and Winther, O. (2018). Recurrent relational networks. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31*, pages 3368–3378. Curran Associates, Inc.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: A method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Parikh, A., Täckström, O., Das, D., and Uszkoreit, J. (2016). A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255, Austin, Texas. Association for Computational Linguistics.

Park, Y., Na, H., Lee, H., Lee, J., and Song, I. (2016). An effective diverse decoding scheme for robust synonymous sentence translation. *AMTA 2016, Vol.*, page 53.

Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In Dasgupta, S. and McAllester, D., editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1310–1318, Atlanta, Georgia, USA. PMLR.

Paulik, M. and Waibel, A. (2009). Automatic translation from parallel speech: Simultaneous interpretation as mt training data. In *2009 IEEE Workshop on Automatic Speech Recognition Understanding*, pages 496–501.

Paulik, M. and Waibel, A. (2013). Training speech translation from audio recordings of interpreter-mediated communication. *Computer Speech & Language*, 27(2):455 – 474. Special Issue on Speech-speech translation.

Paulus, R., Xiong, C., and Socher, R. (2017). A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*.

Pennington, J., Socher, R., and Manning, C. D. (2014). GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Pereyra, G., Tucker, G., Chorowski, J., Kaiser, Ł., and Hinton, G. E. (2017). Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*.

Peter, J.-T., Guta, A., Rossenbach, N., Graça, M., and Ney, H. (2016). The RWTH Aachen machine translation system for IWSLT 2016. In *International Workshop on Spoken Language Translation. Seattle, WA, USA*.

Peters, M., Ammar, W., Bhagavatula, C., and Power, R. (2017). Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1756–1765. Association for Computational Linguistics.

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

Pinnis, M., Krišlauks, R., Deksne, D., and Miks, T. (2017). Neural machine translation for morphologically rich languages with improved sub-word units and synthetic data. In Ekštein, K. and Matoušek, V., editors, *Text, Speech, and Dialogue*, pages 237–245, Cham. Springer International Publishing.

Platanios, E. A., Stretcu, O., Neubig, G., Poczos, B., and Mitchell, T. (2019). Competence-based curriculum learning for neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1162–1172, Minneapolis, Minnesota. Association for Computational Linguistics.

Pollack, J. B. (1990). Recursive distributed representations. *Artificial Intelligence*, 46(1):77 – 105.

Poncelas, A., Shterionov, D., Way, A., Wenniger, G. M. d. B., and Passban, P. (2018). Investigating backtranslation in neural machine translation. *arXiv preprint arXiv:1804.06189*.

Popel, M. and Bojar, O. (2018). Training tips for the Transformer model. *The Prague Bulletin of Mathematical Linguistics*, 110(1):43–70.

Popescu-Belis, A. (2019). Context in neural machine translation: A review of models and evaluations. *arXiv preprint arXiv:1901.09115*.

Post, M. (2018). A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels. Association for Computational Linguistics.

Pouget-Abadie, J., Bahdanau, D., van Merrienboer, B., Cho, K., and Bengio, Y. (2014). Overcoming the curse of sentence length for neural machine translation using automatic segmentation. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 78–85, Doha, Qatar. Association for Computational Linguistics.

Povey, D., Ghoshal, A., and Boulianne, G. (2011). The Kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society.

Powell, M. J. (1964). An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The computer journal*, 7(2):155–162.

Powell, M. J. (2009). The BOBYQA algorithm for bound constrained optimization without derivatives. *Cambridge NA Report NA2009/06, University of Cambridge, Cambridge*.

Prabhavalkar, R., Alsharif, O., Bruguier, A., and McGraw, L. (2016). On the compression of recurrent neural networks with an application to LVCSR acoustic modeling for embedded speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5970–5974.

Pramanik, S. and Hussain, A. (2019). Text normalization using memory augmented neural networks. *Speech Communication*, 109:15 – 23.

Puduppully, R., Zhang, Y., and Shrivastava, M. (2016). Transition-based syntactic linearization with lookahead features. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 488–493, San Diego, California. Association for Computational Linguistics.

Quinn, J. and Ballesteros, M. (2018). Pieces of eight: 8-bit neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers)*, pages 114–120, New Orleans - Louisiana. Association for Computational Linguistics.

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving language understanding with unsupervised learning. Technical report, Technical report, OpenAI.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners. https://openai.com/blog/better-language-models/.

Raffel, C., Luong, M.-T., Liu, P. J., Weiss, R. J., and Eck, D. (2017). Online and linear-time attention by enforcing monotonic alignments. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 2837–2846. JMLR.org.

Ramachandran, P., Liu, P. J., and Le, Q. V. (2017). Unsupervised pretraining for sequence to sequence learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 383–391, Copenhagen, Denmark. Association for Computational Linguistics.

Ranzato, M., Chopra, S., Auli, M., and Zaremba, W. (2015). Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*.

Rarrick, S., Quirk, C., and Lewis, W. D. (2011). MT detection in web-scraped parallel corpora. *Proceedings of the Machine Translation Summit (MT Summit XIII)*.

Ren, S., Chen, W., Liu, S., Li, M., Zhou, M., and Ma, S. (2018). Triangular architecture for rare language translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 56–65, Melbourne, Australia. Association for Computational Linguistics.

Ren, S., Zhang, Z., Liu, S., Zhou, M., and Ma, S. (2019). Unsupervised neural machine translation with SMT as posterior regularization. *arXiv preprint arXiv:1901.04112*.

Resnik, P. (1999). Mining the web for bilingual text. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 527–534, College Park, Maryland, USA. Association for Computational Linguistics.

Resnik, P. and Smith, N. A. (2003). The web as a parallel corpus. *American Journal of Computational Linguistics*, 29(3):349–380.

Ribeiro, M., Singh, S., and Guestrin, C. (2016). "Why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 97–101, San Diego, California. Association for Computational Linguistics.

Rikters, M. (2018). Debugging neural machine translations. *arXiv preprint arXiv:1808.02733*.

Rikters, M. and Bojar, O. (2017). Paying attention to multi-word expressions in neural machine translation. *arXiv preprint arXiv:1710.06313*.

Rikters, M. and Fishel, M. (2017). Confidence through attention. *arXiv preprint arXiv:1710.03743*.

Roark, B., Sproat, R., Allauzen, C., Riley, M. D., Sorensen, J., and Tai, T. (2012). The OpenGrm open-source finite-state grammar software libraries. In *Proceedings of the ACL 2012 System Demonstrations*, pages 61–66. Association for Computational Linguistics.

Rodriguez, T. and Seal, M. (2014). CyHunspell. https://github.com/MSeal/cython_hunspell. [Online; accessed 4-May-2019].

Rokach, L. (2010). Ensemble-based classifiers. *Artificial Intelligence Review*, 33(1):1–39.

Rossenbach, N., Rosendahl, J., Kim, Y., Graça, M., Gokrani, A., and Ney, H. (2018). The RWTH Aachen University filtering system for the WMT 2018 parallel corpus filtering task. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 946–954, Belgium, Brussels. Association for Computational Linguistics.

Ruiz, N., Gangi, M. A. D., Bertoldi, N., and Federico, M. (2017). Assessing the tolerance of neural machine translation systems against speech recognition errors. In *Proc. Interspeech 2017*, pages 2635–2639.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1988). *Neurocomputing: Foundations of Research*, chapter Learning Representations by Back-propagating Errors, pages 696–699. MIT Press, Cambridge, MA, USA.

Rush, A. M., Chopra, S., and Weston, J. (2015). A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal. Association for Computational Linguistics.

Russell, S. J., Norvig, P., Canny, J. F., Malik, J. M., and Edwards, D. D. (2003). *Artificial intelligence: A modern approach*, volume 2. Prentice hall Upper Saddle River.

Sajjad, H., Durrani, N., Dalvi, F., Belinkov, Y., and Vogel, S. (2017). Neural machine translation training in a multi-domain scenario. In *International Workshop on Spoken Language Translation*.

Sakaguchi, K., Post, M., and van Durme, B. (2017). Grammatical error correction with neural reinforcement learning. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 366–372, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Salesky, E., Runge, A., Coda, A., Niehues, J., and Neubig, G. (2018). Optimizing segmentation granularity for neural machine translation. *arXiv preprint arXiv:1810.08641*.

Sankaran, B., Freitag, M., and Al-Onaizan, Y. (2017). Attention-based vocabulary selection for NMT decoding. *arXiv preprint arXiv:1706.03824*.

Santoro, A., Raposo, D., Barrett, D. G., Malinowski, M., Pascanu, R., Battaglia, P., and Lillicrap, T. (2017). A simple neural network module for relational reasoning. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 4967–4976. Curran Associates, Inc.

Saunders, D., de Gispert, A., Stahlberg, F., and Byrne, B. (2019). Domain adaptive inference for neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics.

Saunders, D., Stahlberg, F., de Gispert, A., and Byrne, B. (2018). Multi-representation ensembles and delayed SGD updates improve syntax-based NMT. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 319–325, Melbourne, Australia. Association for Computational Linguistics.

Schmaltz, A., Kim, Y., Rush, A. M., and Shieber, S. (2017). Adapting sequence models for sentence correction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2807–2813. Association for Computational Linguistics.

Schmaltz, A., Rush, A. M., and Shieber, S. (2016). Word ordering without syntax. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2319–2324, Austin, Texas. Association for Computational Linguistics.

Schmidt, T. and Marg, L. (2018). How to move to neural machine translation for enterprise-scale programs—an early adoption case study.

Schnober, C., Eger, S., Do Dinh, E.-L., and Gurevych, I. (2016). Still not there? Comparing traditional sequence-to-sequence models to encoder-decoder neural networks on monotone string translation tasks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1703–1714, Osaka, Japan. The COLING 2016 Organizing Committee.

Schuster, M. and Nakajima, K. (2012). Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152.

Schuster, M. and Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Schwarzenberg, R., Harbecke, D., Macketanz, V., Avramidis, E., and Möller, S. (2019). Train, sort, explain: Learning to diagnose translation models. *arXiv preprint arXiv:1903.12017*.

Schwenk, H. (2008). Investigations on large-scale lightly-supervised training for statistical machine translation. In *International Workshop on Spoken Language Translation (IWSLT) 2008*, pages 182–189.

Schwenk, H. (2012). Continuous space translation models for phrase-based statistical machine translation. In *Proceedings of COLING 2012: Posters*, pages 1071–1080, Mumbai, India. The COLING 2012 Organizing Committee.

Schwenk, H. (2014). Université du Maine. http://www-lium.univ-lemans.fr/~schwenk/nnmt-shared-task/. [Online; accessed 4-May-2016].

Schwenk, H., Dechelotte, D., and Gauvain, J.-L. (2006). Continuous space language models for statistical machine translation. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 723–730, Sydney, Australia. Association for Computational Linguistics.

Sculley, D., Snoek, J., Wiltschko, A., and Rahimi, A. (2018). Winner's curse? On pace, progress, and empirical rigor. *openreview.net*.

SDL (2018). SDL cracks Russian to English neural machine translation. https://www.sdl.com/about/news-media/press/2018/sdl-cracks-russian-to-english-neural-machine-translation.html.

See, A., Luong, M.-T., and Manning, C. D. (2016). Compression of neural machine translation models via pruning. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 291–301, Berlin, Germany. Association for Computational Linguistics.

Sennrich, R. (2017). How grammatical is character-level neural machine translation? Assessing MT quality with contrastive translation pairs. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 376–382, Valencia, Spain. Association for Computational Linguistics.

Sennrich, R., Birch, A., Currey, A., Germann, U., Haddow, B., Heafield, K., Miceli Barone, A. V., and Williams, P. (2017a). The University of Edinburgh's neural MT systems for WMT17. In *Proceedings of the Second Conference on Machine Translation*, pages 389–399, Copenhagen, Denmark. Association for Computational Linguistics.

Sennrich, R., Firat, O., Cho, K., Birch, A., Haddow, B., Hitschler, J., Junczys-Dowmunt, M., Läubli, S., Miceli Barone, A. V., Mokry, J., and Nadejde, M. (2017b). Nematus: A toolkit for neural machine translation. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 65–68, Valencia, Spain. Association for Computational Linguistics.

Sennrich, R. and Haddow, B. (2016). Linguistic input features improve neural machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 83–91, Berlin, Germany. Association for Computational Linguistics.

Sennrich, R., Haddow, B., and Birch, A. (2016a). Edinburgh neural machine translation systems for WMT 16. In *Proceedings of the First Conference on Machine Translation*, pages 371–376, Berlin, Germany. Association for Computational Linguistics.

Sennrich, R., Haddow, B., and Birch, A. (2016b). Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Sennrich, R., Haddow, B., and Birch, A. (2016c). Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Serrano, S. and Smith, N. A. (2019). Is attention interpretable? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Florence, Italy. Association for Computational Linguistics.

Shah, H. and Barber, D. (2018). Generative neural machine translation. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31*, pages 1346–1355. Curran Associates, Inc.

Shang, L., Lu, Z., and Li, H. (2015). Neural responding machine for short-text conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1577–1586, Beijing, China. Association for Computational Linguistics.

Shaw, P., Uszkoreit, J., and Vaswani, A. (2018). Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468, New Orleans, Louisiana. Association for Computational Linguistics.

Shen, S., Cheng, Y., He, Z., He, W., Wu, H., Sun, M., and Liu, Y. (2016). Minimum risk training for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1683–1692, Berlin, Germany. Association for Computational Linguistics.

Shen, T., Zhou, T., Long, G., Jiang, J., Pan, S., and Zhang, C. (2018a). DiSAN: Directional self-attention network for RNN/CNN-free language understanding. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Shen, T., Zhou, T., Long, G., Jiang, J., Wang, S., and Zhang, C. (2018b). Reinforced self-attention network: A hybrid of hard and soft attention for sequence modeling. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, pages 4345–4352. AAAI Press.

Shen, T., Zhou, T., Long, G., Jiang, J., Wang, S., and Zhang, C. (2018c). Reinforced self-attention network: A hybrid of hard and soft attention for sequence modeling. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, pages 4345–4352. AAAI Press.

Shen, Y., Tan, S., Sordoni, A., and Courville, A. (2019). Ordered neurons: Integrating tree structures into recurrent neural networks. *Proceedings of ICLR*.

Shuyo, N. (2010). Language detection library for Java. http://code.google.com/p/language-detection/. [Online; accessed 1-June-2016].

Siahbani, M., Sankaran, B., and Sarkar, A. (2013). Efficient left-to-right hierarchical phrase-based translation with improved reordering. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1089–1099, Seattle, Washington, USA. Association for Computational Linguistics.

Siegelmann, H. T. and Sontag, E. D. (1995). On the computational power of neural nets. *Journal of Computer and System Sciences*, 50(1):132–150.

Sim, K. C., Byrne, B., Gales, M. J., Sahbi, H., and Woodland, P. C. (2007). Consensus network decoding for statistical machine translation system combination. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*, volume 4, pages IV–105–IV–108.

Skorokhodov, I., Rykachevskiy, A., Emelyanenko, D., Slotin, S., and Ponkratov, A. (2018). Semi-supervised neural machine translation with language models. In *Proceedings of the AMTA 2018 Workshop on Technologies for MT of Low Resource Languages (LoResMT 2018)*, pages 37–44, Boston, MA. Association for Machine Translation in the Americas.

Smith, S. L., Kindermans, P.-J., Ying, C., and Le, Q. V. (2017). Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489*.

So, D. R., Liang, C., and Le, Q. V. (2019). The evolved Transformer. *arXiv preprint arXiv:1901.11117*.

Socher, R., Pennington, J., Huang, E. H., Ng, A. Y., and Manning, C. D. (2011). Semi-supervised recursive Autoencoders for predicting sentiment distributions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 151–161, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Sønderby, S. K., Sønderby, C. K., Nielsen, H., and Winther, O. (2015). Convolutional LSTM networks for subcellular localization of proteins. In Dediu, A.-H., Hernández-Quiroz, F., Martín-Vide, C., and Rosenblueth, D. A., editors, *Algorithms for Computational Biology*, pages 68–80, Cham. Springer International Publishing.

Song, K., Xu, T., Peng, F., and Lu, J. (2018). Hybrid self-attention network for machine translation. *arXiv preprint arXiv:1811.00253*.

Sountsov, P. and Sarawagi, S. (2016). Length bias in encoder decoder models and a case for global conditioning. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1516–1525, Austin, Texas. Association for Computational Linguistics.

Specia, L. (2011). Exploiting objective annotations for measuring translation post-editing effort. In *Proceedings of the 15th Conference of the European Association for Machine Translation*, pages 73–80.

Specia, L., Blain, F., Logacheva, V., Astudillo, R., and Martins, A. F. T. (2018). Findings of the WMT 2018 shared task on quality estimation. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 689–709, Belgium, Brussels. Association for Computational Linguistics.

Specia, L., Frank, S., Sima'an, K., and Elliott, D. (2016). A shared task on multimodal machine translation and crosslingual image description. In *Proceedings of the First Conference on Machine Translation*, pages 543–553, Berlin, Germany. Association for Computational Linguistics.

Sperber, M., Neubig, G., Niehues, J., and Waibel, A. (2017). Neural lattice-to-sequence models for uncertain inputs. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1380–1389, Copenhagen, Denmark. Association for Computational Linguistics.

Sproat, R. and Jaitly, N. (2016). RNN approaches to text normalization: A challenge. *arXiv preprint arXiv:1611.00068*.

Srinivas, S. and Babu, R. V. (2015). Data-free parameter pruning for deep neural networks. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 31.1–31.12. BMVA Press.

Sriram, A., Jun, H., Satheesh, S., and Coates, A. (2018). Cold fusion: Training seq2seq models together with language models. In *Proc. Interspeech 2018*, pages 387–391.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Stahlberg, F., Bryant, C., and Byrne, B. (2019a). Neural grammatical error correction with finite state transducers. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4033–4039, Minneapolis, Minnesota. Association for Computational Linguistics.

Stahlberg, F. and Byrne, B. (2017). Unfolding and shrinking neural machine translation ensembles. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1946–1956, Copenhagen, Denmark. Association for Computational Linguistics.

Stahlberg, F. and Byrne, B. (2019a). The CUED's grammatical error correction systems for BEA19. In *Proceedings of the 14th ACL-HLT Workshop on Innovative Use of NLP for Building Educational Applications*, Florence, Italy. Association for Computational Linguistics.

Stahlberg, F. and Byrne, B. (2019b). On NMT search errors and model errors: Cat got your tongue? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, Hong Kong. Association for Computational Linguistics.

Stahlberg, F., Cross, J., and Stoyanov, V. (2018a). Simple fusion: Return of the language model. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 204–211, Belgium, Brussels. Association for Computational Linguistics.

Stahlberg, F., de Gispert, A., and Byrne, B. (2018b). The University of Cambridge's machine translation systems for WMT18. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 504–512, Belgium, Brussels. Association for Computational Linguistics.

Stahlberg, F., de Gispert, A., Hasler, E., and Byrne, B. (2017a). Neural machine translation by minimising the Bayes-risk with respect to syntactic translation lattices. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 362–368, Valencia, Spain. Association for Computational Linguistics.

Stahlberg, F., Hasler, E., and Byrne, B. (2016a). The edit distance transducer in action: The University of Cambridge English-German system at WMT16. In *Proceedings of the First Conference on Machine Translation*, pages 377–384, Berlin, Germany. Association for Computational Linguistics.

Stahlberg, F., Hasler, E., Saunders, D., and Byrne, B. (2017b). SGNMT – a flexible NMT decoding platform for quick prototyping of new models and search strategies. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 25–30, Copenhagen, Denmark. Association for Computational Linguistics.

Stahlberg, F., Hasler, E., Waite, A., and Byrne, B. (2016b). Syntactically guided neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 299–305, Berlin, Germany. Association for Computational Linguistics.

Stahlberg, F., Saunders, D., and Byrne, B. (2018c). An operation sequence model for explainable neural machine translation. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 175–186, Brussels, Belgium. Association for Computational Linguistics.

Stahlberg, F., Saunders, D., de Gispert, A., and Byrne, B. (2019b). CUED@WMT19:EWC&LMs. In *Proceedings of the Fourth Conference on Machine Translation: Shared Task Papers*. Association for Computational Linguistics.

Stahlberg, F., Saunders, D., Iglesias, G., and Byrne, B. (2018d). Why not be versatile? Applications of the SGNMT decoder for machine translation. In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Papers)*, pages 208–216, Boston, MA. Association for Machine Translation in the Americas.

Stahlberg, F. and Vogel, S. (2015). The QCRI recognition system for handwritten Arabic. In Murino, V. and Puppo, E., editors, *Image Analysis and Processing — ICIAP 2015*, pages 276–286, Cham. Springer International Publishing.

Stenetorp, P. (2013). Transition-based dependency parsing using recursive neural networks. In *NIPS Workshop on Deep Learning*. Citeseer.

Stern, M., Chan, W., Kiros, J. R., and Uszkoreit, J. (2019). Insertion Transformer: Flexible sequence generation via insertion operations. *arXiv preprint arXiv:1902.03249*.

Stern, M., Shazeer, N., and Uszkoreit, J. (2018). Blockwise parallel decoding for deep autoregressive models. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31*, pages 10086–10095. Curran Associates, Inc.

Stolcke, A. (2002). SRILM – an extensible language modeling toolkit. In *Seventh international conference on spoken language processing*.

Su, J., Tan, Z., Xiong, D., Ji, R., Shi, X., and Liu, Y. (2017). Lattice-based recurrent neural network encoders for neural machine translation. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, pages 3302–3308. AAAI Press.

Su, J., Wu, S., Xiong, D., Lu, Y., Han, X., and Zhang, B. (2018). Variational recurrent neural machine translation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Sukhbaatar, S., Szlam, A., Weston, J., and Fergus, R. (2015). End-to-end memory networks. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28*, pages 2440–2448. Curran Associates, Inc.

Sun, G.-Z., Chen, H.-H., Giles, C. L., Lee, Y.-C., and Chen, D. (1990). Connectionist pushdown automata that learn context-free grammars. In *Proceedings of the International Joint Conference on Neural Networks*, volume 1, pages 577–580. Lawrence Earlbaum Hillsdale, NJ.

Sun, G.-Z., Giles, C. L., Chen, H.-H., and Lee, Y.-C. (1993). The neural network pushdown automation: Model, stack and learning simulations. Technical report, University of Maryland at College Park, College Park, MD, USA.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Swietojanski, P. and Renals, S. (2014). Learning hidden unit contributions for unsupervised speaker adaptation of neural network acoustic models. In *2014 IEEE Spoken Language Technology Workshop (SLT)*, pages 171–176.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

Taghipour, K., Khadivi, S., and Xu, J. (2011). Parallel corpus refinement as an outlier detection algorithm. *Proceedings of the 13th Machine Translation Summit (MT Summit XIII)*, pages 414–421.

Tai, K. S., Socher, R., and Manning, C. D. (2015). Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, Beijing, China. Association for Computational Linguistics.

Takase, S. and Okazaki, N. (2019). Positional encoding to control output sequence length. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3999–4004, Minneapolis, Minnesota. Association for Computational Linguistics.

Tamchyna, A., Weller-Di Marco, M., and Fraser, A. (2017). Modeling target-side inflection in neural machine translation. In *Proceedings of the Second Conference on Machine Translation*, pages 32–42, Copenhagen, Denmark. Association for Computational Linguistics.

Tan, Z., Su, J., Wang, B., Chen, Y., and Shi, X. (2018). Lattice-to-sequence attentional neural machine translation models. *Neurocomputing*, 284:138 – 147.

Tang, G., Müller, M., Rios, A., and Sennrich, R. (2018a). Why self-attention? A targeted evaluation of neural machine translation architectures. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4263–4272, Brussels, Belgium. Association for Computational Linguistics.

Tang, G., Sennrich, R., and Nivre, J. (2018b). An analysis of attention mechanisms: The case of word sense disambiguation in neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 26–35, Belgium, Brussels. Association for Computational Linguistics.

Tang, Y., Meng, F., Lu, Z., Li, H., and Yu, P. L. (2016). Neural machine translation with external phrase memory. *arXiv preprint arXiv:1606.01792*.

Tars, S. and Fishel, M. (2018). Multi-domain neural machine translation. In *Proceedings of the 21st Annual Conference of the European Association for Machine Translation: 28-30 May 2018, Universitat d'Alacant, Alacant, Spain*, pages 259–268. European Association for Machine Translation.

Thompson, B., Gwinnup, J., Khayrallah, H., Duh, K., and Koehn, P. (2019). Overcoming catastrophic forgetting during domain adaptation of neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2062–2068, Minneapolis, Minnesota. Association for Computational Linguistics.

Thompson, B., Khayrallah, H., Anastasopoulos, A., McCarthy, A. D., Duh, K., Marvin, R., McNamee, P., Gwinnup, J., Anderson, T., and Koehn, P. (2018). Freezing subnetworks to analyze domain adaptation in neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 124–132, Belgium, Brussels. Association for Computational Linguistics.

Tiedemann, J. and Scherrer, Y. (2017). Neural machine translation with extended context. In *Proceedings of the Third Workshop on Discourse in Machine Translation*, pages 82–92, Copenhagen, Denmark. Association for Computational Linguistics.

Tomasello, L. (2019). *Neural Machine Translation and Artificial Intelligence: What Is Left for the Human Translator?* PhD thesis, University of Padua.

Tomczak, M. (2016). Bachbot. MPhil dissertation, University of Cambridge.

Tong, A., Diduch, L., Fiscus, J., Haghpanah, Y., Huang, S., Joy, D., Peterson, K., and Soboroff, I. (2018). Overview of the NIST 2016 LoReHLT evaluation. *Machine Translation*, 32(1-2):11–30.

Toral, A., Castilho, S., Hu, K., and Way, A. (2018). Attaining the unattainable? Reassessing claims of human parity in neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 113–123, Belgium, Brussels. Association for Computational Linguistics.

Toral, A. and Sánchez-Cartagena, V. M. (2017). A multifaceted evaluation of neural versus phrase-based machine translation for 9 language directions. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1063–1073, Valencia, Spain. Association for Computational Linguistics.

Tran, K., Bisazza, A., and Monz, C. (2018). The importance of being recurrent for modeling hierarchical structure. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4731–4736, Brussels, Belgium. Association for Computational Linguistics.

Tromble, R., Kumar, S., Och, F. J., and Macherey, W. (2008). Lattice Minimum Bayes-Risk decoding for statistical machine translation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 620–629, Honolulu, Hawaii. Association for Computational Linguistics.

Tu, Z., Liu, Y., Shang, L., Liu, X., and Li, H. (2017). Neural machine translation with reconstruction. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, pages 3097–3103. AAAI Press.

Tu, Z., Liu, Y., Shi, S., and Zhang, T. (2018). Learning to remember translation history with a continuous cache. *Transactions of the Association for Computational Linguistics*, 6:407–420.

Tu, Z., Lu, Z., Liu, Y., Liu, X., and Li, H. (2016). Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 76–85, Berlin, Germany. Association for Computational Linguistics.

Ueffing, N. and Ney, H. (2005). Word-level confidence estimation for machine translation using phrase-based translation models. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 763–770, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Upadhyay, S., Faruqui, M., Dyer, C., and Roth, D. (2016). Cross-lingual models of word embeddings: An empirical comparison. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1661–1670, Berlin, Germany. Association for Computational Linguistics.

Vaibhav, V., Singh, S., Stewart, C., and Neubig, G. (2019). Improving robustness of machine translation with synthetic noise. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1916–1920, Minneapolis, Minnesota. Association for Computational Linguistics.

van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. (2016a). WaveNet: A generative model for raw audio. In *9th ISCA Speech Synthesis Workshop*, pages 125–125.

van den Oord, A., Kalchbrenner, N., and Kavukcuoglu, K. (2016b). Pixel recurrent neural networks. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 1747–1756. JMLR.org.

van der Wees, M., Bisazza, A., and Monz, C. (2017). Dynamic data selection for neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1400–1410, Copenhagen, Denmark. Association for Computational Linguistics.

van Merriënboer, B., Bahdanau, D., Dumoulin, V., Serdyuk, D., Warde-Farley, D., Chorowski, J., and Bengio, Y. (2015). Blocks and fuel: Frameworks for deep learning. *arXiv preprint arXiv:1506.00619*.

Vaswani, A., Bengio, S., Brevdo, E., Chollet, F., Gomez, A. N., Gouws, S., Jones, L., Kaiser, Ł., Kalchbrenner, N., Parmar, N., Sepassi, R., Shazeer, N., and Uszkoreit, J. (2018). Tensor2Tensor for neural machine translation. In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Papers)*, pages 193–199, Boston, MA. Association for Machine Translation in the Americas.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Vaswani, A., Zhao, Y., Fossum, V., and Chiang, D. (2013). Decoding with large-scale neural language models improves translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1387–1392, Seattle, Washington, USA. Association for Computational Linguistics.

Vijayakumar, A. K., Cogswell, M., Selvaraju, R. R., Sun, Q., Lee, S., Crandall, D., and Batra, D. (2016). Diverse beam search: Decoding diverse solutions from neural sequence models. *arXiv preprint arXiv:1610.02424*.

Vilar, D. (2018). Learning hidden unit contribution for adapting neural machine translation models. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 500–505, New Orleans, Louisiana. Association for Computational Linguistics.

Vinyals, O., Kaiser, Ł., Koo, T., Petrov, S., Sutskever, I., and Hinton, G. E. (2015). Grammar as a foreign language. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28*, pages 2773–2781. Curran Associates, Inc.

Vogel, S., Ney, H., and Tillmann, C. (1996). HMM-based word alignment in statistical translation. In *COLING 1996 Volume 2: The 16th International Conference on Computational Linguistics*.

Voita, E., Serdyukov, P., Sennrich, R., and Titov, I. (2018). Context-aware neural machine translation learns anaphora resolution. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1264–1274, Melbourne, Australia. Association for Computational Linguistics.

Volkart, L., Bouillon, P., and Girletti, S. (2018). Statistical vs. neural machine translation: A comparison of mth and deepl at swiss post's language service. In *Proceedings of the 40th Conference Translating and the Computer*, pages 145–150, London, United-Kingdom.

Vu, T., Hu, B., Munkhdalai, T., and Yu, H. (2018). Sentence simplification with memory-augmented neural networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 79–85, New Orleans, Louisiana. Association for Computational Linguistics.

Waibel, A., Hanazawa, T., Hinton, G. E., Shikano, K., and Lang, K. J. (1989). Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(3):328–339.

Wang, C., Li, M., and Smola, A. (2019a). Language models with Transformers. *arXiv preprint arXiv:1904.09408*.

Wang, C., Zhang, J., and Chen, H. (2018a). Semi-autoregressive neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 479–488, Brussels, Belgium. Association for Computational Linguistics.

Wang, L., Tu, Z., Way, A., and Liu, Q. (2017a). Exploiting cross-sentence context for neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2826–2831, Copenhagen, Denmark. Association for Computational Linguistics.

Wang, M., Gong, L., Zhu, W., Xie, J., and Bian, C. (2018b). Tencent neural machine translation systems for WMT18. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 522–527, Belgium, Brussels. Association for Computational Linguistics.

Wang, M., Lu, Z., Li, H., and Liu, Q. (2016). Memory-enhanced decoder for neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 278–286, Austin, Texas. Association for Computational Linguistics.

Wang, R., Finch, A., Utiyama, M., and Sumita, E. (2017b). Sentence embedding for neural machine translation domain adaptation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 560–566, Vancouver, Canada. Association for Computational Linguistics.

Wang, R., Utiyama, M., Finch, A., Liu, L., Chen, K., and Sumita, E. (2018c). Sentence selection and weighting for neural machine translation domain adaptation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(10):1727–1741.

Wang, R., Utiyama, M., Liu, L., Chen, K., and Sumita, E. (2017c). Instance weighting for neural machine translation domain adaptation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1482–1488, Copenhagen, Denmark. Association for Computational Linguistics.

Wang, W., Watanabe, T., Hughes, M., Nakagawa, T., and Chelba, C. (2018d). Denoising neural machine translation training with trusted data and online data selection. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 133–143, Belgium, Brussels. Association for Computational Linguistics.

Wang, X., Lu, Z., Tu, Z., Li, H., Xiong, D., and Zhang, M. (2017d). Neural machine translation advised by statistical machine translation. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, pages 3330–3336. AAAI Press.

Wang, X., Pham, H., Dai, Z., and Neubig, G. (2018e). SwitchOut: An efficient data augmentation algorithm for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 856–861, Brussels, Belgium. Association for Computational Linguistics.

Wang, X., Pham, H., Yin, P., and Neubig, G. (2018f). A tree-based decoder for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4772–4777, Brussels, Belgium. Association for Computational Linguistics.

Wang, X., Tu, Z., and Zhang, M. (2018g). Incorporating statistical machine translation word knowledge into neural machine translation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(12):2255–2266.

Wang, X., Utiyama, M., and Sumita, E. (2018h). CytonMT: An efficient neural machine translation open-source toolkit implemented in C++. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 133–138, Brussels, Belgium. Association for Computational Linguistics.

Wang, Y., Cheng, S., Jiang, L., Yang, J., Chen, W., Li, M., Shi, L., Wang, Y., and Yang, H. (2017e). Sogou neural machine translation systems for WMT17. In *Proceedings of the Second Conference on Machine Translation*, pages 410–415, Copenhagen, Denmark. Association for Computational Linguistics.

Wang, Y., Tian, F., He, D., Qin, T., Zhai, C., and Liu, T.-Y. (2019b). Non-autoregressive machine translation with auxiliary regularization. *arXiv preprint arXiv:1902.10245*.

Wang, Y., Xia, Y., Zhao, L., Bian, J., Qin, T., Liu, G., and Liu, T.-Y. (2018i). Dual transfer learning for neural machine translation with marginal distribution regularization. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Wang, Z. (2018). Simultaneous neural machine translation. 4-th year project, University of Cambridge.

Welleck, S., Brantley, K., Daumé III, H., and Cho, K. (2019). Non-monotonic sequential text generation. *arXiv preprint arXiv:1902.02192*.

Werlen, L. M., Pappas, N., Ram, D., and Popescu-Belis, A. (2018). Global-context neural machine translation through target-side attentive residual connections. *researchgate.net*.

Wieting, J. and Kiela, D. (2019). No training required: Exploring random encoders for sentence classification. *arXiv preprint arXiv:1901.10444*.

Williams, A., Drozdov, A., and Bowman, S. (2018). Do latent tree learning models identify meaningful structure in sentences? *Transactions of the Association for Computational Linguistics*, 6:253–267.

Williams, P., Sennrich, R., Post, M., and Koehn, P. (2016). Syntax-based statistical machine translation. *Synthesis Lectures on Human Language Technologies*, 9(4):1–208.

Williams, R. J. and Zipser, D. (1989). A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280.

Wiseman, S. and Rush, A. M. (2016). Sequence-to-sequence learning as beam-search optimization. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1296–1306, Austin, Texas. Association for Computational Linguistics.

Wong, J. H. and Gales, M. J. (2016). Sequence student-teacher training of deep neural networks. In *Interspeech 2016*, pages 2761–2765.

Woods, W. A. (1970). Transition network grammars for natural language analysis. *Commun. ACM*, 13(10):591–606.

Wu, D. (1997). Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403.

Wu, F., Fan, A., Baevski, A., Dauphin, Y. N., and Auli, M. (2019a). Pay less attention with lightweight and dynamic convolutions. In *ICLR*.

Wu, J., Leng, C., Wang, Y., Hu, Q., and Cheng, J. (2016a). Quantized convolutional neural networks for mobile devices. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4820–4828.

Wu, J., Wang, X., and Wang, W. Y. (2019b). Extract and edit: An alternative to back-translation for unsupervised neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1173–1183, Minneapolis, Minnesota. Association for Computational Linguistics.

Wu, L., Tian, F., Qin, T., Lai, J., and Liu, T.-Y. (2018a). A study of reinforcement learning for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3612–3621, Brussels, Belgium. Association for Computational Linguistics.

Wu, L., Xia, Y., Zhao, L., Tian, F., Qin, T., Lai, J., and Liu, T.-Y. (2017a). Adversarial neural machine translation. *arXiv preprint arXiv:1704.06933*.

Wu, S., Zhang, D., Yang, N., Li, M., and Zhou, M. (2017b). Sequence-to-dependency neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 698–707, Vancouver, Canada. Association for Computational Linguistics.

Wu, W., Wang, H., Liu, T., and Ma, S. (2018b). Phrase-level self-attention networks for universal sentence encoding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3729–3738, Brussels, Belgium. Association for Computational Linguistics.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. (2016b). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Wu, Y. and Zhao, H. (2018). Finding better subword segmentation for neural machine translation. In Sun, M., Liu, T., Wang, X., Liu, Z., and Liu, Y., editors, *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*, pages 53–64, Cham. Springer International Publishing.

Xia, Y., Qin, T., Chen, W., Bian, J., Yu, N., and Liu, T.-Y. (2017). Dual supervised learning. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 3789–3798. JMLR.org.

Xie, Z., Avati, A., Arivazhagan, N., Jurafsky, D., and Ng, A. Y. (2016). Neural language correction with character-based attention. *arXiv preprint arXiv:1603.09727*.

Xiong, H., He, Z., Hu, X., and Wu, H. (2018a). Multi-channel encoder for neural machine translation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Xiong, H., He, Z., Wu, H., and Wang, H. (2018b). Modeling coherence for discourse neural machine translation. *arXiv preprint arXiv:1811.05683*.

Xu, H. and Liu, Q. (2019). Neutron: An implementation of the Transformer translation model and its variants. *arXiv preprint arXiv:1903.07402*.

Xu, K., Ba, J. L., Kiros, J. R., Cho, K., Courville, A., Salakhutdinov, R., Zemel, R., and Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057.

Xu, X., Kuang, S., and Xiong, D. (2018). Two effective approaches to data reduction for neural machine translation: Static and dynamic sentence selection. In *2018 International Conference on Asian Language Processing (IALP)*, pages 159–164.

Xue, J., Li, J., and Gong, Y. (2013). Restructuring of deep neural network acoustic models with singular value decomposition. In *Interspeech*, pages 2365–2369.

Yang, B., Wong, D. F., Xiao, T., Chao, L. S., and Zhu, J. (2017a). Towards bidirectional hierarchical representations for attention-based neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1432–1441, Copenhagen, Denmark. Association for Computational Linguistics.

Yang, J., Zhang, B., Qin, Y., Zhang, X., Lin, Q., and Su, J. (2018a). Otem&utem: Over- and under-translation evaluation metric for nmt. In Zhang, M., Ng, V., Zhao, D., Li, S., and Zan, H., editors, *Natural Language Processing and Chinese Computing*, pages 291–302, Cham. Springer International Publishing.

Yang, Y., Huang, L., and Ma, M. (2018b). Breaking the beam search curse: A study of (re-)scoring methods and stopping criteria for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3054–3059, Brussels, Belgium. Association for Computational Linguistics.

Yang, Z., Chen, L., and Le Nguyen, M. (2018c). Regularizing forward and backward decoding to improve neural machine translation. In *2018 10th International Conference on Knowledge and Systems Engineering (KSE)*, pages 73–78.

Yang, Z., Chen, W., Wang, F., and Xu, B. (2016a). A character-aware encoder for neural machine translation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3063–3070, Osaka, Japan. The COLING 2016 Organizing Committee.

Yang, Z., Chen, W., Wang, F., and Xu, B. (2018d). Improving neural machine translation with conditional sequence generative adversarial nets. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1346–1355, New Orleans, Louisiana. Association for Computational Linguistics.

Yang, Z., He, X., Gao, J., Deng, L., and Smola, A. (2016b). Stacked attention networks for image question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 21–29.

Yang, Z., Hu, Z., Deng, Y., Dyer, C., and Smola, A. (2017b). Neural machine translation with recurrent attention modeling. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 383–387, Valencia, Spain. Association for Computational Linguistics.

Yannakoudakis, H., Rei, M., Andersen, Ø. E., and Yuan, Z. (2017). Neural sequence-labelling models for grammatical error correction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2795–2806, Copenhagen, Denmark. Association for Computational Linguistics.

Yao, L., Torabi, A., Cho, K., Ballas, N., Pal, C., Larochelle, H., and Courville, A. (2015). Describing videos by exploiting temporal structure. In *Proceedings of the IEEE international conference on computer vision*, pages 4507–4515.

Yu, L., Blunsom, P., Dyer, C., Grefenstette, E., and Kocisky, T. (2016a). The neural noisy channel. *arXiv preprint arXiv:1611.02554*.

Yu, L., Buys, J., and Blunsom, P. (2016b). Online segment to segment neural transduction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1307–1316, Austin, Texas. Association for Computational Linguistics.

Yu, L., d'Autume, C. d. M., Dyer, C., Blunsom, P., Kong, L., and Ling, W. (2018). Sentence encoding with tree-constrained relation networks. *arXiv preprint arXiv:1811.10475*.

Yu, L., Zhang, W., Wang, J., and Yu, Y. (2017). SeqGAN: Sequence generative adversarial nets with policy gradient. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, pages 2852–2858. AAAI Press.

Yuan, Z. and Briscoe, T. (2016). Grammatical error correction using neural machine translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 380–386, San Diego, California. Association for Computational Linguistics.

Yuan, Z., Stahlberg, F., Rei, M., Byrne, B., and Yannakoudakis, H. (2019). Neural and FST-based approaches to grammatical error correction. In *Proceedings of the 14th ACL-HLT Workshop on Innovative Use of NLP for Building Educational Applications*, Florence, Italy. Association for Computational Linguistics.

Zamora-Martinez, F., Castro-Bleda, M. J., and Schwenk, H. (2010). N-gram-based machine translation enhanced with neural networks for the French-English BTEC-IWSLT'10 task. In *International Workshop on Spoken Language Translation (IWSLT) 2010*.

Zaremba, W., Sutskever, I., and Vinyals, O. (2014). Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Zaremoodi, P. and Haffari, G. (2018). Incorporating syntactic uncertainty in neural machine translation with a forest-to-sequence model. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1421–1429, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Zeiler, M. D. (2012). ADADELTA: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Zenkel, T., Wuebker, J., and DeNero, J. (2019). Adding interpretable attention to neural translation models improves word alignment. *arXiv preprint arXiv:1901.11359*.

Zhang, B., Xiong, D., and Su, J. (2017a). A gru-gated attention model for neural machine translation. *arXiv preprint arXiv:1704.08430*.

Zhang, B., Xiong, D., Su, J., Duan, H., and Zhang, M. (2016). Variational neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 521–530, Austin, Texas. Association for Computational Linguistics.

Zhang, D., Crego, J., and Senellart, J. (2018a). Analyzing knowledge distillation in neural machine translation. In *International Workshop on Spoken Language Translation IWSLT*.

Zhang, D., Kim, J., Crego, J., and Senellart, J. (2017b). Boosting neural machine translation. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 271–276, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Zhang, H., Sproat, R., Ng, A. H., Stahlberg, F., Peng, X., Gorman, K., and Roark, B. (2019a). Neural models of text normalization for speech applications. *Computational Linguistics*, 0(0):1–45.

Zhang, J., Ding, Y., Shen, S., Cheng, Y., Sun, M., Luan, H., and Liu, Y. (2017c). THUMT: An open source toolkit for neural machine translation. *arXiv preprint arXiv:1706.06415*.

Zhang, J., Luan, H., Sun, M., Zhai, F., Xu, J., Zhang, M., and Liu, Y. (2018b). Improving the Transformer translation model with document-level context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 533–542, Brussels, Belgium. Association for Computational Linguistics.

Zhang, J., Utiyama, M., Sumita, E., Neubig, G., and Nakamura, S. (2017d). Improving neural machine translation through phrase-based forced decoding. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 152–162, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Zhang, J., Utiyama, M., Sumita, E., Neubig, G., and Nakamura, S. (2018c). Guiding neural machine translation with retrieved translation pieces. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1325–1335, New Orleans, Louisiana. Association for Computational Linguistics.

Zhang, J. and Zong, C. (2016a). Bridging neural machine translation and bilingual dictionaries. *arXiv preprint arXiv:1610.07272*.

Zhang, J. and Zong, C. (2016b). Exploiting source-side monolingual data in neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545, Austin, Texas. Association for Computational Linguistics.

Zhang, Q., Liang, S., and Yilmaz, E. (2018d). Variational self-attention model for sentence representation. *arXiv preprint arXiv:1812.11559*.

Zhang, W. E., Sheng, Q. Z., and Alhazmi, A. A. F. (2019b). Generating textual adversarial examples for deep learning models: A survey. *arXiv preprint arXiv:1901.06796*.

Zhang, X., Su, J., Qin, Y., Liu, Y., Ji, R., and Wang, H. (2018e). Asynchronous bidirectional decoding for neural machine translation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Zhang, Y., Blackwood, G., and Clark, S. (2012). Syntax-based word ordering incorporating a large-scale language model. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 736–746, Avignon, France. Association for Computational Linguistics.

Zhang, Y. and Clark, S. (2011). Syntax-based grammaticality improvement using CCG and guided search. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1147–1157, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Zhang, Y. and Clark, S. (2015). Discriminative syntax-based word ordering for text generation. *Computational Linguistics*, 41(3):503–538.

Zhang, Z., Liu, S., Li, M., Zhou, M., and Chen, E. (2018f). Bidirectional generative adversarial networks for neural machine translation. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 190–199, Brussels, Belgium. Association for Computational Linguistics.

Zhang, Z., Liu, S., Li, M., Zhou, M., and Chen, E. (2018g). Joint training for neural machine translation models with monolingual data. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Zhang, Z., Wang, R., Utiyama, M., Sumita, E., and Zhao, H. (2018h). Exploring recombination for efficient decoding of neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4785–4790, Brussels, Belgium. Association for Computational Linguistics.

Zhang, Z., Wu, S., Liu, S., Li, M., Zhou, M., and Chen, E. (2018i). Regularizing neural machine translation by target-bidirectional agreement. *arXiv preprint arXiv:1808.04064*.

Zhao, W., Wang, L., Shen, K., Jia, R., and Liu, J. (2019). Improving grammatical error correction via pre-training a copy-augmented architecture with unlabeled data. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 156–165, Minneapolis, Minnesota. Association for Computational Linguistics.

Zhou, L., Hu, W., Zhang, J., and Zong, C. (2017). Neural system combination for machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 378–384, Vancouver, Canada. Association for Computational Linguistics.

Zhu, M. and Gupta, S. (2017). To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*.

Zhukov, V., Golikov, E., and Kretov, M. (2017). Differentiable lower bound for expected bleu score. *arXiv preprint arXiv:1712.04708*.

Zipf, G. K. (1946). The psychology of language. In *Encyclopedia of psychology*, pages 332–341. Philosophical Library.

Zoph, B. and Knight, K. (2016). Multi-source neural translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 30–34, San Diego, California. Association for Computational Linguistics.

Zoph, B. and Le, Q. V. (2016). Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*.

Zoph, B., Yuret, D., May, J., and Knight, K. (2016). Transfer learning for low-resource neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1568–1575, Austin, Texas. Association for Computational Linguistics.

From my rotting body, flowers shall grow and I am in them and that is eternity.

Edvard Munch

A

List of Publications

The contributions of this thesis draw from the following publications I authored or co-authored in the course of my PhD studies:

- Stahlberg et al. (2016a): **Stahlberg, F.**, Hasler, E., and Byrne, B. (2016a). The edit distance transducer in action: The University of Cambridge English-German system at WMT16. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 377–384. Association for Computational Linguistics.
- Stahlberg et al. (2016b): **Stahlberg, F.**, Hasler, E., Waite, A., and Byrne, B. (2016b). Syntactically guided neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)* , pages 299–305. Association for Computational Linguistics.
- Stahlberg et al. (2017a): **Stahlberg, F.**, de Gispert, A., Hasler, E., and Byrne, B. (2017a). Neural machine translation by minimising the Bayes-risk with respect to syntactic translation lattices. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 362–368. Association for Computational Linguistics.
- Stahlberg et al. (2017b): **Stahlberg, F.**, Hasler, E., Saunders, D., and Byrne, B. (2017b). SGNMT – A flexible NMT decoding platform for quick prototyping of new models

and search strategies. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 25–30. Association for Computational Linguistics.

- Stahlberg and Byrne (2017): **Stahlberg, F.** and Byrne, B. (2017). Unfolding and shrinking neural machine translation ensembles. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1946–1956. Association for Computational Linguistics.

- Hasler et al. (2017a): Hasler, E., de Gispert, A., **Stahlberg, F.**, Waite, A., and Byrne, B. (2017a). Source sentence simplification for statistical machine translation. *Computer Speech & Language*, 45:221 – 235.

- Hasler et al. (2017b): Hasler, E., **Stahlberg, F.**, Tomalin, M., de Gispert, A., and Byrne, B. (2017b). A comparison of neural models for word ordering. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 208–212. Association for Computational Linguistics.

- Stahlberg et al. (2018a): **Stahlberg, F.**, Cross, J., and Stoyanov, V. (2018a). Simple fusion: Return of the language model. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 204–211. Association for Computational Linguistics.

- Stahlberg et al. (2018b): **Stahlberg, F.**, de Gispert, A., and Byrne, B. (2018b). The University of Cambridge's machine translation systems for WMT18. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 504–512. Association for Computational Linguistics.

- Stahlberg et al. (2018c): **Stahlberg, F.**, Saunders, D., and Byrne, B. (2018c). An operation sequence model for explainable neural machine translation. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 175–186. Association for Computational Linguistics.

- Stahlberg et al. (2018d): **Stahlberg, F.**, Saunders, D., Iglesias, G., and Byrne, B. (2018d). Why not be versatile? Applications of the SGNMT decoder for machine translation. In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Papers)*, pages 208–216. Association for Machine Translation in the Americas.

- Saunders et al. (2018): Saunders, D., **Stahlberg, F.**, de Gispert, A., and Byrne, B. (2018). Multi-representation ensembles and delayed SGD updates improve syntax-based NMT. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 319–325. Association for Computational Linguistics.

- Stahlberg et al. (2019a): **Stahlberg, F.**, Bryant, C., and Byrne, B. (2019). Neural grammatical error correction with finite state transducers. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

- Zhang et al. (2019a): Zhang, H., Sproat, R., Ng, A. H., **Stahlberg, F.**, Peng, X., Gorman, K., and Roark, B. (2019). Neural models of text normalization for speech applications. *Computational Linguistics*.

- Stahlberg and Byrne (2019a): **Stahlberg, F.** and Byrne, B. (2019). The CUED's grammatical error correction systems for BEA19. In *Proceedings of the 14th ACL-HLT Workshop on Innovative Use of NLP for Building Educational Applications*, Florence, Italy. Association for Computational Linguistics.

- Yuan et al. (2019): Yuan, Z., **Stahlberg, F.**, Rei, M., Byrne, B., and Yannakoudakis, H. (2019). Neural and FST-based approaches to grammatical error correction. In *Proceedings of the 14th ACL-HLT Workshop on Innovative Use of NLP for Building Educational Applications*, Florence, Italy. Association for Computational Linguistics.

- Stahlberg et al. (2019b): **Stahlberg, F.**, Saunders, D., de Gispert, A., and Byrne, B. (2019b). CUED@WMT19:EWC&LMs. In *Proceedings of the Fourth Conference on Machine Translation: Shared Task Papers*. Association for Computational Linguistics.

- Saunders et al. (2019): Saunders, D., **Stahlberg, F.**, de Gispert, A., and Byrne, B. (2019). Domain adaptive inference for neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics.

- Stahlberg and Byrne (2019b): **Stahlberg, F.** and Byrne, B (2019b). On NMT search errors and model errors: Cat got your tongue? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, Hong Kong. Association for Computational Linguistics.

Index

A* search, 137
Activation function, 24, 157
Active hypothesis, 40
Actor network, 46
Adadelta, 61, 159
Adagrad, 158
Adam, 116
Adaptation, 43, 75, **76**, 77
Adequacy, 60, 67, 87, 146
Admissible pruning, 137, 141
Adversarial training, 68
Alignment error rate, 225
Alignments, 202, 214, 215, 223, 224
  Monotonic alignments, 71, 73
  Soft alignment, 27, 214, 225
  Word alignments, 9, 70, 216
Ambiguity, 46, 52, 70, 80, 82
Annotations, 31, 41
Attention, **26**, 30, 37, 50, 51, 70, 72, 78, 80, 82, 88, 93, 152, 171, 225
  Additive, 27
  Attention weight matrix, 27, 29, 214
  Dot-product, 27
  Hard attention, 71, 73
  Masked attention, **30**, 36, 72, 196
  Multi-head attention, **27**, 36, 37, 72
  Scaled dot-product, 27
  Scoring function, 26
  Self-attention, 35, 72
Attention heads, 27
Autoencoders, 22, 56
  Recurrent autoencoders, 21

Back-translation, **56**, 66, 67, 75–78, 114, 170, 189
Backpropagation, 61, 64, 68
Bag of words, 21, 170
Bag2seq, 171–173
Batch decoding, 138, 209
Batch normalization, 64
Batch size, 48, 66, 79, 117
Bayes rule, 8
BEA-2019 shared task, 182
Beam decoding, **39**, 41, 45, 91, 107, 113, 124, 132, 136, 138, 139, 148, 172, 175, 178
Beam size, 46, 57, 101, 183
Blocks, 93, 124, 138, 148
BOBYQA, 109
Bottom-up, 52
Byte pair encoding, **52**, 115, 160
  Iterative BPE, 54

Calibration, 58, 70
Catastrophic errors, 2, **204**, 211, 213
Catastrophic forgetting, 76, 149
Chain rule, 8, 30
Channels, 33

Character-based NMT, **51**, 54, 61
Checkpoint averaging, **44**, 117
CKY algorithm, 16
Closed vocabulary, **48**, 49, 51, 80, 93
Clustering, 46
Cold fusion, 186, 189
Commercial MT, 51, 106, 124
Composition, **14**, 174, 175, 177, 183
Compound-splitting, 51
Compression, 52, 80
Computer vision, 32, 65, 68, 80
Confidence, 70
Confusion sets, **175**, 180
Constrained decoding, 71, 76, 88, **91**, 144, 174, 205, 211, 213, 223
Content-based addressing, 36
Context, 21, 27, 35, 36, 45, 72, **80**, 204, 207, 208, 210
  Document-level context, 82, 87, 170, 195
Contextual sequence-to-sequence models, 205, 206
Continued training, 76, 77
Continuous optimization, 45
Convolution, 116
  Depthwise convolution, 33
  Depthwise separable convolution, 34
  Pointwise convolution, 33
Convolutional neural networks, 22, **32**, 51, 184
ConvS2S, 35, **37**, 61
Coverage model, 60
  Linguistic coverage model, 60
  Neural network based coverage model, 60
Coverage penalty, 60, 109
Coverage vector, 60
CPU, 43, 45
Cross-entropy, **62**, 65, 67
Curriculum learning, 76

Data selection, 75, 114
Data-bound shrinking, 158
Decoder, 23, 24, 26, 27, 30
Decoder state, 24, 27, 30, 31, 35, 41, 69
Decoders, 124, **133**
Decoding speed, 43, 45, 46, 72, 126, 138, 144, 151, 162, 166, 170, 205, 209, 211
Decoupling, 124, 134
Deep fusion, 55, 185
Deep learning, 1, 2, 64
Delayed SGD, **117**, 222
Deletions, 182, 184
Depth-first search, 94, **136**, 137, 140, 218
Derivation, 15, 90, 220
Determinization, **14**, 90, 96, 109, 178
Dirac distribution, 63, 65
Disambiguation, 100
Distortion model, 10, 88
Diversity, 46, 87, 105, 147
Document-level language modelling, 195
Document-level NMT, 82, 195
Domain, 45, 51, **76**, 86
Dropout, 65, 77, 116
Dual learning
  Dual supervised learning, 68
  Dual unsupervised learning, 56

Early stopping, 65
Edit distance, 86, 97, 175
  Edit distance transducer, 98
Elastic weight consolidation, 77, 195
Embedding matrix, 21, 48, 152, 159
Empty translation, 139, 143
Encoder, 25, 27, 30, 31
Encoder-decoder networks, **23**, 26, 37, 74, 206

End-of-sentence symbol, 24, 39, 41
End-to-end training, 1, 2
Ensembling, 3, **42**, 61, 97, 102, 105, 106, 109, 113, 118, 126, 130, 151, 153, 160, 170, 175, 179, 183, 193
  Multi-representation ensembles, 203
Entropy, 193
Evidence space, 106, 113
Exact search, 139
Exhaustive search, 91, 106, 139
Explainability, 69, 214, 215
Exposure bias, 67
Extensibility, 126, 148

Factored machine translation, 82
Feedback loop, 24, 88, 159
Feedforward neural network, 19, 24, 31, 60, 153, 164, 205, 208
Fertility, **10**, 60, 71, 72, 88
FFcontext textnorm model, 205, **208**, 210
Filtering, 75, 76, 114
  Dual filtering, 76
Fine-tuning, **76**, 78
Finite state machines, 11
Fixed-length representation, **23**, 26, 30
Flower automaton, 98
Fluence, 170
Fluency, 55, 60, 70, 87, 170, 193
Fred Jelinek, 1

Garden-path problem, 39, 40
Gated activation, 24, 51, 73, 88
Generalization, 64, 76
Generative adversarial networks, 68
Generative models, 9, 169
Generative story, 9, 10
Giza++, 222, 223, 225
GNMT, 2, **37**, 61
Golden-section search, 148
GPU, 34, 43, 45, 48, 66, 79, 117, 138, 152, 162, 222
Gradient-based optimization, 21, 37, 44, 48, 59, 61, 161
Grammars, 15, 55, 90, 215
  Chomsky grammars, 201
  Context-free grammars, 15, 16, 220
  Covering grammars, 205, **213**
  Inversion transduction grammars, 22, 74
  Multitext grammars, 202, **218**
  Recurrent neural network grammars, 81
  Synchronous grammars, 15, 218
Grammatical error correction, 4, 149, 170, **174**, 182
Greedy decoding, **39**, 41, 46, 58, 91, 132, 136, 137
GRU, 24, 31, 93, 157, 159, 161, 205, 207
Gyro, 173

Hebb's learning rule, 155
Heuristics, 137, 172, 173
Hierarchical phrase-based machine translation, 15
Hiero, 3, **15**, 51, 55, 57, 85, 88, 169, 201, 215, 220
HiFST, 16, 90, 99, 109
High cohesion, 134
Huffman codes, 52
Hybrid systems, **85**
Hypothesis expansion, 39
Hypothesis recombination, 46
Hypothesis space, 106

IBM Model 1, 10
IBM Model 3, 10

IBM models, 9, 55, 216
Importance sampling, 50
Informed search, 137
Insertion-based translation, 45, 77
Insertions, 16, 45, 77, 100, 103, 183
Instance weighting, 76
Interpretability, 69
  Post-hoc interpretability, 69
Intra-inter Transformer, 196, 199

Kernel, 33
Knowledge distillation, 43, 46, 66, 77, **80**, 149, 152
Knowledge graphs, 82

L2 regularization, **65**, 77
Label smoothing, 58, **65**, 116, 189
Language detection, 114
Language models, 8, 21, 50, 55, 72, 86, 93, 127, **169**, 177
  Neural language models, 8, 75, 192
  Recurrent neural network language models, 171, 173, 192
Latency, 47
Lattice rescoring, 130
Lattice size, 96, 112
Lattice-based NMT, 81
Lattices, 19, 50, 76, 87, 88, 99, 103, 105
Layer normalization, 64, 116
Lecture translation, 47
Left-to-right factorization, **23**, 30, 39, 45, 89, 106, 107, 113, 139, 207
Length normalization, **59**, 118, 141, 145
Lexical choice, 82
Lexical translation probability, 10, 87
Line search, 148
Linear combination, 73, 130, 146, 152, 157, 158, 178
Linearization, 81, 170
Linearized parse trees, 203
Linguistically-motivated subword units, 53
Location-based addressing, 36
Log semiring, 13
Log-linear models, **11**, 16, 20, 55, 87
Logits, 155
Long-range dependencies, 36
Loose combination, 88, 97, 105
LoReHLT, 77
Low rank matrix factorization, 80, 159
Low-resource machine translation, 77, 87, 194
LSTM, 21, 24, 31, 51, 81, 116, 118

Maximum likelihood training, 57, 62, 67
Maximum mutual information, 47
Maxout layer, 32, 93, 159, 161, 163
MBR-based NMT, 86, **105**, **107**, 112, 146, 148, 150, 195, 199
Memory, 26, 27, 36, 72, **74**, 79, 82, 117, 153
Minimization, **14**, 90, 96, 109, 178
Minimum Bayes-risk decoding, 45, 46, 105, **106**, 112
Minimum error rate training, 59, 94, 99, 109, 113, 147
Minimum risk training, 67
Model errors, 57, 58, 96, 139
Monolingual data, **55**, 76, 77, 185, 189
Monte Carlo estimation, 63
Morphology, 21, 51, 53, 69, 79, 175, 204
Multi-pass decoding, 45
Multi-representation ensembles, 81
Multi-source NMT, 44, 45, **79**, 82
Multi-task learning, 56
Multilevel textnorm model, 205, **206**, 210

Multilingual NMT, 51, 77, **78**
Multimodal machine translation, 80
Music composition, 130, 149

Natural language processing, 11, 20, 21, 23, 33, 35, 48, 68, 69
Neural architecture search, 79
Neural joint models, 19
Neural language models, 8, 19
Neural stack, 74
Neural Turing machine, 74
Neuron activity, 44, 46, 64, 65, 68, 69, 80, 155, 158, 161, 166
NMT toolkits, 125
Noise, 46, 52, 54, 56, 66, 68, 75, 87
Noise-contrastive estimation, 50
Noisy-channel model, 8
Non-autoregressive neural machine translation, 74
Normalization, 90
NPLM, 94, 124
Numerical stability, 43

OOV, 49–51, 90, 93, 94, 98, 108
OOV rate, 48
OpenFST, 100, 102, 109, 124, 149
Operation sequence model (SMT), 215, 216, **221**
Operation sequence neural machine translation, 215
Optical character recognition, 12
OSNMT, 202
Output formats, 146
Over-fitting, **64**, 76
Over-parameterization, 64, 79, 152, 163
Over-sampling, 56, 114
Over-translation, **60**, 72, 86

Padding, **29**, 33
Parallelization, 34, 45, 66, 74, 79
Parameter sharing, 78
Partial hypothesis, 39, 41
Perplexity, 193, 195, 199
Phrase embeddings, 21
Phrase table, **10**, 50, 87, 88
Phrase-based translation, 10, 55, 70, 88
  Phrase-based NMT, 73
Pivot-based translation, 78
Positional encodings, **36**, 37, 61, 72, 171
  Relative positional encodings, 116, 118
Post-editing, 70, 87
Post-processing, 88, 111
PostNorm, 186
Powell's method, 113, **148**
Pre-training, 21, 57, 76
Precision, 79
Predictors, 124, 137, 148
  Predictor state, 127
  Predictor weights, 147
  Predictor wrappers, 131, 203
PreNorm, 186
Preprocessing, 51, 92, 96, 115, 188
Projection function, **5**, 12, **15**, 41, 100, 102, 176, 213
Projection layer, 37
Pruning, 79, 101

Quality estimation, 70
Quantization, 79

Rare words, 48, 51
Read head, 216
Real-time translation, 47
Recurrent autoencoders, 21
Recurrent continuous translation models, 23

Recursive transition networks, 16
Regularization, 45, 58, **65**, 75, 76, 88
Reinforcement learning, 45, 47, **67**
Relation networks, 23
Reranking, 88, 197
Rescoring, 88, 91, 97, 106, 109, 114
Reshuffling, 94
Residual connections, 37, 64
Residual probability, 187
Reward, 59, 67
Right-to-left decoding, 44, 114, 118
Risk, 105–107
RNMT+, 37
RNN, 24, 36, 46, 51, 73
  Bi-scale RNN, 51
  Bidirectional RNNs, 25, **31**
  Unidirectional RNNs, 25
RNNsearch, 30, 32, **41**, 48, 112, 116, 132, 153, 171, 188, 204, 205
RNNsearch-LV, **50**
Robustness, 48, 54, 75, 76
Rule-based MT, 2, 202

SDL, 2, 124, 150
Search errors, 57, 58, 96, **139**
Search space, 86, 88, 91, 106, 107, 111, 134, 136, 139, 174, 181
Segmentation, 47, 206
Self-normalizing training, 50
Semiring, 12, 14, 90
  Tropical semiring, 90, 178
  Tropical sparse tuple vector semiring, 99
Sentence length, 26, 57, 87
SentencePiece model, 52
Sentiment analysis, 21–23, 35
SGNMT, 86, 118, **123**, 160
Shallow fusion, 55, 185, 186, 189, 195
Shortest path search, 40, 100, 134, 136
Shrinking, 79, 152
  Data-bound shrinking, 152
  Data-free shrinking, 152
Simple fusion, 185
Simultaneous translation, **47**, 67, 71, 149
Singular value decomposition, 80, 152, 159
SliceNet, **35**, 116, 118
Sliding window, 205, 210
Softmax, 24, 26, 32, 37, 43, 48, 50, 90, 155
  Local softmax, 96
Source context, 19, 26
Space invariance, 33
Spatial dimension, 33
Spatial invariance, 33
Speech recognition, 12, 29, 80, 81
Speech-to-speech translation, 1, 47
Spell checking, 175
Spurious ambiguity, 222
Statistical machine translation, 2, **7**, 45, 50, 55, 70, 75–77, 82, 85, 87, 118, 169, 173, 174, 179, 195, 203, 214
Subword regularization, 52
Subword units, **52**, 54, 73, 80, 108, 111, 130, 149, 178, 188, 205, 206, 222
Syntactic constituency trees, 202
Syntactically guided NMT, 76, 86, 88
Syntax, 55, 81, 202
Syntax-based NMT, 203
System combination, 88

Teaching, 149
Tensor2Tensor, 116, 117, 124, 126, 148
TensorFlow, 116, 124, 126
Text normalization, 2, 4, 29, 74, **204**
Text-to-speech, 1, 204, 205
Theano, 93, 124, 126, 148

Tokenization, 51, 52, 81, 115, 132, 149, 183, 205–207, 210
Training, 44
Transfer learning, 77
Transformer, 21, 29, 36, **37**, 57, 58, 61, 64, 70, 71, 79, 82, 112, 117, 143, 179, 184, 195, 196, 216, 222
  Relative Transformer, 116, 118
Translation memory, 47
Translation prefix, 35, 39, 41, 45, 47, 74, 90, 135, 178
Translatotron, 1
Travatar, 109
Tree-based NMT, 81
Tree-LSTMs, 81
Trees, 15, 22, 45, 52, 92, 218
  Binary trees, 22, 45, 220
  Dependency trees, 81
  Lineraized trees, 81
  Syntactic constituency trees, 81
  Tree-structured search space, 91, 136
Tropical semiring, 13
Tuning, 146

UML, 133
Under-translation, **60**, 72, 86
Unfolding, 152, **153**
Unigram, 50, 172, 183
UNK, **48**, 49, 50, 90, 91, 94, 98–100, 103, 106, 107, 111, 173, 211
UNK insertion, 100, 103
UNK replace, 49, 50, 94, 111
UNK substitution, 98
Unsupervised learning, 56, 77
Unsupervised NMT, 57, **77**, 88

Vanishing gradient problem, 24, 61, 64
Variational neural machine translation, 74
Visualization, 69
Vocabulary selection, 50
Vocabulary size, 48, 51, 52, 80, 86, 90, 96, 104, 109, 205

WAT, 109, 160
WaveNet, 1
Weight matrix, 154
Weight pushing, **14**, 90, 91, 96
Weighted finite state automaton, 14
Weighted finite state transducers, 11, 46, 90, 97, 108, 134, 174, 182, 203, 211
WMT, 43, 44, 50, 55, 56, 70, 75, 77, 78, 80, 86, 88, 93, 94, 97, 109, 112, 138, 143, 151, 160, 172, 188, 197
Word alignments, 9
Word embeddings, **21**, 21, 36, 48, 78
  Contextualized word embeddings, 21, 72
  Cross-lingual word embeddings, 21, 77
Word order, 36, **170**, 170, 227
Word penalty, 59
Word sense disambiguation, 70
Word-based NMT, 48
Wordpiece model, 52
Write head, 216
Writing system, 55

Zero-shot translation, 78
Zipf's law, 48

Author Index

Abadi, Martín 29, 124, 128, 148, 198
Abel, Andrew 74
Adam, Hartwig 34
Adel, Heike 29
Agapiou, John 74
Agirre, Eneko 77
Aharoni, Roee 71, 78, 81, 203, 229
Ahmed, Karim 72
Ahn, Sungjin 49, 74
Aho, Alfred V. 15
Akhanov, Egor 20
Akoury, Nader 74, 81
Al-Onaizan, Yaser 43, 44, 46, 50, 70, 76, 78–80
Alhazmi, Ahoud Abdulrahmn F 69
Alishahi, Afra 69
Alkhouli, Tamer 71
Allauzen, Alexandre 19
Allauzen, Cyril 12, 14, 16, 109, 124, 128, 146, 149, 175, 177, 213
Alsharif, Ouais 80
Alvarez-Melis, David 2, 69
Álvaro Peris 126
Ammar, Waleed 21
Amodei, Dario 66, 72
Anastasopoulos, Antonios 29, 76
Andersen, Øistein E. 185
Anderson, Tim 76
Andreas, Jacob 50
Andreetto, Marco 34
Andrychowicz, Marcin 74
Arase, Yuki 75
Arčan, Mihael 82
Arivazhagan, Naveen 174
Artetxe, Mikel 77
Arthur, Philip 88, 126
Astudillo, Ramón 70
Ataman, Duygu 53
Atkinson, Kevin 175
Attardo, Donalee 97
Aue, Anthony 2, 56, 68
Augasta, M. Gethsiyal 79
Auli, Michael xviii, 34, 35, 37, 38, 50, 56, 61, 64, 66, 67, 70, 79, 126, 139, 220
Avati, Anand 174
Avramidis, Eleftherios 2, 69, 88, 112, 120, 121
Axelrod, Amittai 75, 76
Aziz, Wilker 74, 81

Ba, Jimmy Lei 29, 37, 64, 71, 116
Babaeizadeh, Mohammad 80, 158
Babu, R. Venkatesh 80, 152, 155–158, 161
Bach, Nguyen 70

Bach, Sebastian 69
Baevski, Alexei 35, 126
Bahar, Parnia 74
Bahdanau, Dzmitry xvii, xxiii, 19, 20, 24–27, 29–32, 41, 48, 61, 72, 92, 93, 116, 124, 126, 128, 132, 134, 138, 148, 149, 152, 157, 159, 171
Ballas, Nicolas 29
Ballesteros, Miguel 79, 81, 229
Bandyopadhyay, Sivaji 87
Banga, Eduardo R. 12, 16, 93, 99, 110
Bangalore, Srinivas 12
Bapna, Ankur 36–38, 54, 72, 112
Barber, David 74
Barham, Paul 29, 124, 128, 148, 198
Barone, Antonio Valerio Miceli 87
Baroni, Marco 23
Barrault, Loïc 23, 82
Barrett, David G 23
Basho xvii, 29
Bastien, Frédéric 29, 93, 124, 126, 128, 148, 159
Bastings, Joost 74, 81, 82
Batra, Dhruv 46, 47
Battaglia, Peter 23
Bau, Anthony 69, 70
Bawden, Rachel 79, 82
Beale, Stephen 97
Beck, Daniel 164
Bekal, Dhanush 82
Belinkov, Yonatan 54, 69, 70, 75, 76
Bellegarda, Jerome R. 20
Bengio, Samy 45, 67, 74, 114, 116, 117, 124–126, 128, 141, 148, 149, 179, 198, 222
Bengio, Yoshua xvii, xxiii, xxvi, 8, 19–21, 24–27, 29–32, 35, 36, 41, 44, 46, 48–51, 54, 55, 59, 61, 64, 68, 71, 72, 74, 76, 78, 87, 92–95, 116, 124, 126, 128, 132, 134, 138, 141, 145, 148, 149, 151, 152, 157, 159, 171, 185, 186, 190, 192, 195
Bentivogli, Luisa 78, 87
Benzeghiba, Mohamed Faouzi 12
Berant, Jonathan 71
Berg-Kirkpatrick, Taylor 46
Bergeron, Arnaud 29, 93, 124, 126, 128, 148, 159
Bergstra, James 29, 93, 124, 126, 128, 148, 159
Bertoldi, Nicola 75, 87
Bhagavatula, Chandra 21
Bhardwaj, Vikas 78
Bhattacharyya, Pushpak 52, 77
Bian, Chao 43, 45
Bian, Jiang 56, 68
Binder, Alexander 69
Birch, Alexandra xxv, 42–45, 52, 56, 61, 76, 79, 81, 82, 102, 105, 114–116, 118, 126, 132, 141, 149, 151, 159, 178, 179, 185, 188, 189, 198, 203, 222
Bird, Steven 29
Bisazza, Arianna 71, 75, 76, 87
Bisk, Yonatan 54, 75
Black, Alan W 51
Blackwood, Graeme 12, 16, 70, 93, 99, 105–108, 110, 170
Blain, Frédéric 70
Bluche, Théodore 12
Blunsom, Phil xvii, 19, 22–24, 29, 71, 74
Bojar, Ondřej 43, 44, 48, 53, 54, 56, 66, 71, 73, 77, 78, 112, 117, 170, 200, 223
Bordes, Antoine 23
Boser, Bernhard 32

Bottou, Léon 21, 32
Bouchard, Nicolas 29, 93, 124, 126, 128, 148, 159
Bougares, Fethi xxvi, 19, 24, 25, 30, 31, 55, 61, 80, 82, 87, 93, 152, 157, 159, 185, 186, 190, 195
Bouillon, Pierrette 87
Boulanger-Lewandowski, Nicolas 59, 141, 145
Boulianne, Gilles 12
Bourque, Pierre 134
Bowman, Samuel 21, 46, 81
Boyd-Graber, Jordan 47, 69
Bradbury, James 21, 74, 81
Brantley, Kianté 45
Brants, Thorsten 177, 179, 185, 209
Bretschner, Gabriel 71
Brevdo, Eugene 114, 116, 117, 124–126, 128, 141, 148, 149, 179, 198, 222
Briscoe, Ted 29, 174–176, 178–180, 183
Britz, Denny 76
Brix, Christopher 74
Brockett, Chris 46
Brown, Peter F. 8–10, 55, 71, 88, 220
Bruguier, Antoine 80
Bruna, Joan 80
Brunelle, Patrice 20
Brunner, Gino 215
Bryant, Christopher 4, 124, 169, 175, 176, 178–180, 183, 301
Buciluǎ, Cristian 80
Buitelaar, Paul 82
Burchardt, Aljoscha 88, 112, 120, 121
Burget, Lukáš 186, 192
Burlot, Franck 56
Buys, Jan 71
Byrne, Bill xx, xxvi, 3, 4, 10, 12, 16, 43–45, 50, 57, 64–66, 70, 71, 77, 81, 85–89, 93, 97, 99, 100, 105–108, 110, 112, 113, 117, 120, 121, 123, 124, 126, 128, 130, 132, 138, 139, 141, 146, 149–151, 164, 169, 170, 172, 173, 179, 182–185, 195, 197–199, 201–203, 223, 299–301

Caccia, Lucas 69
Caccia, Massimo 69
Calixto, Iacer 80, 87
Campbell, Roy H. 80, 158
Canny, John F 134, 136, 137
Cao, Yuan 2, 20, 37, 38, 42, 46, 52, 59, 60, 67, 79, 109, 118, 141, 151, 170
Carbonell, Jaime 72
Carl, Michael 87
Carpuat, Marine 56, 75
Caruana, Rich 80
Casacuberta, Francisco 51, 126
Case, Carl 126
Cashman, Dylan 69
Castilho, Sheila 2, 87
Castro-Bleda, Maria José 19
Černockỳ, Jan 186, 192
Cettolo, Mauro 78, 87
Chan, William 29, 45
Chandar, Sarath 74
Chang, Ming-Wei 21, 72
Chang, Remco 69
Chao, Lidia S. 81
Charlin, Laurent 69
Chatterjee, Rajen 43, 54, 56, 77
Che, Wanxiang 170
Chelba, Ciprian 76, 177, 179, 209
Chen, Bo 34
Chen, Boxing 76

Chen, Chang 2, 56, 68
Chen, D 74
Chen, Enhong 45, 56, 68, 69
Chen, Haiqing 2, 74
Chen, Hsing-Hen 74
Chen, Huadong 81
Chen, Jiajun 81
Chen, Jianmin 29, 124, 128, 148, 198
Chen, Kai 21
Chen, Kehai 76, 81
Chen, Laifu 45
Chen, Mia Xu 36–38, 72, 112
Chen, Wei 43, 45, 51, 55, 68, 69, 170
Chen, Wenhu 71, 77
Chen, Xie 46
Chen, Xinlei 2, 69
Chen, Yidong 81
Chen, Yining 140
Chen, Yufeng 49
Chen, Yun 46
Chen, Zhehuai 12
Chen, Zhibo 74
Chen, Zhifeng 2, 20, 29, 36–38, 42, 46, 52, 59, 60, 67, 72, 78, 79, 109, 112, 118, 124, 128, 141, 148, 151, 170, 198
Cheng, Jian 79
Cheng, Jianpeng 35
Cheng, Shanbo 43, 45, 55, 170
Cheng, Yong 56, 67, 68, 72, 75, 78, 126
Cherry, Colin 54, 76, 87
Chiang, David xviii, 3, 7, 8, 15, 16, 29, 48, 50, 51, 55, 58, 59, 70, 77, 81, 85, 86, 88, 94, 97, 124, 125, 128, 131, 141, 169, 201, 218, 230, 233
Child, Rewon 72
Chitnis, Rohan 52
Chiu, Chung-Cheng 71
Cho, Eunah 47, 57, 88, 139
Cho, Kyunghyun xvii, xxiii, xxvi, 19–21, 24–27, 29–32, 41, 44–51, 54, 55, 59, 61, 71, 72, 74, 77–79, 81, 82, 87, 92–95, 116, 126, 128, 132, 134, 141, 145, 151, 152, 157, 159, 171, 185, 186, 190, 195
Cho, Sungzoon 29
Choi, Eunsol 71
Choi, Gyu-Hyeon 79
Choi, Heeyoul 21, 72
Choi, Young Sang 80
Chollampatt, Shamil 174
Chollet, Francois xxiv, 34, 35, 114, 116–118, 124–126, 128, 141, 143, 148, 149, 179, 198, 222
Chomsky, Noam 201
Chopra, Sumit 29, 64, 67
Chorowski, Jan 29, 65, 66, 93, 124, 126, 128, 134, 138, 148, 149, 159
Chowdhary, Vishal 2, 56, 68
Chowdhury, Koel Dutta 87
Christian, Federmann 78
Chrupala, Grzegorz 69
Chu, Chenhui 42, 43, 76, 79, 151
Chung, Junyoung 51, 54, 55, 151
Clark, Christopher 21
Clark, Jonathan H. 2, 8, 48, 56, 68, 93, 128, 131, 177
Clark, Stephen 81, 170
Clifton, Ann 126
Clopath, Claudia 77
Coates, Adam xxvi, 185, 186, 190
Cocke, John 8
Coda, Alex 52, 54
Cogswell, Michael 47

Cohen, William W 72
Cohn, Trevor 29, 45, 56, 71, 88
Collobert, Ronan 21, 46, 76
Colmenarejo, Sergio Gómez 74
Conneau, Alexis 23, 77, 78
Coquard, Aurelien 20
Cormen, Thomas H 136
Corrado, Greg 21, 78
Costa-Jussá, Marta R 61
Cotterell, Ryan 79
Courville, Aaron 29, 32, 44, 68, 71, 76, 81, 93, 159
Courville, Aaron C 68
Crandall, David 47
Crego, Josep 20, 76, 80
Cromieres, Fabien 20, 42, 43, 151
Cross, James 4, 169, 185, 300
Cui, Lei 75
Currey, Anna 43, 56, 79, 81, 114, 116, 118

Dabre, Raj 20, 76, 77, 79
Dahlmann, Leonard 88
Dahlmeier, Daniel 179
Dai, Zihang 56, 66, 72
Dakwale, Praveen 76
Dally, William 79
Dalvi, Fahim 42, 43, 54, 69, 70, 76, 151, 229
Danihelka, Ivo 74
Daniil, Gavrilov 36
Das, Dipanjan 36
Das, Dipankar 87
Daumé III, Hal 45, 47
Dauphin, Yann N. xviii, 34, 35, 37, 38, 46
d'Autume, Cyprien de Masson 23
Davis, Andy 29, 124, 128, 148, 198
de Freitas, Nando 80
de Gispert, Adrià xxvi, 3, 4, 10, 12, 16, 43–45, 64, 66, 70, 71, 77, 81, 85, 86, 93, 99, 100, 105–108, 110, 112, 117, 120, 121, 124, 128, 130, 141, 146, 149, 150, 164, 169, 170, 172, 173, 179, 195, 197–199, 202, 203, 223, 299–301
Dean, Jeffrey 21, 29, 78, 80, 124, 128, 148, 185, 198
Dechelotte, Daniel 8, 19
Deksne, Daiga 53
Della Pietra, Stephen A. 8–10, 55, 71, 88, 220
Della Pietra, Vincent J. 8–10, 55, 71, 88, 220
DeNero, John 52, 71
Deng, Li 29, 73
Deng, Yongchao 20
Deng, Yuntian 72, 126
Denil, Misha 80
Denker, John S. 79, 152
Denkowski, Michael 56, 126
Denoyer, Ludovic 77, 78
Denton, Emily L 80
Desjardins, Guillaume 77
Devin, Matthieu 29, 124, 128, 148, 198
Devlin, Jacob 19, 21, 50, 72, 79
Dhanuka, Nishikant 20
Di Gangi, Mattia Antonino 45
Diduch, Lukas 77
Dieleman, Sander 1
Dietterich, Thomas G. 42, 151
Ding, Yanzhuo 2, 69, 126, 214, 215
Dinh, Laurent 80
Do Dinh, Erik-Lân 87
Dolan, Bill 46
Domashnev, Constantine 97
Domhan, Tobias 56, 72, 126
Domingo, Miguel 51

Dong, Daxiang 78
Dong, Li 29, 35
dos Santos, Cicero 23
Doshi-Velez, Finale 2, 69
Dowling, Meghan 87
Drozdov, Andrew 81
Du, Jinhua 88
Duan, Hong 74
Ducharme, Réjean 8, 19, 20, 192
Duchi, John 158
Duh, Kevin 76, 77, 88, 109
Dumoulin, Vincent 93, 124, 126, 128, 134, 138, 148, 149, 159
Duong, Long 29
Durrani, Nadir 42, 43, 54, 69, 70, 76, 151, 215, 217, 221, 229
Dutil, Francis 51
Dwojak, Tomasz 44, 79, 81, 87, 88, 117, 126, 138, 203
Dyer, Chris 21, 23, 46, 48, 50, 51, 71, 72, 80, 81, 88, 229

Eck, Douglas 71
Eck, Matthias 76
Edunov, Sergey 56, 64, 66, 67, 79, 126
Edwards, Douglas D 134, 136, 137
Eger, Steffen 87
Eisenstein, Jacob 75
Elliott, Desmond 80
ElMaghraby, Ayah 45
Elman, Jeffrey L 170
Emelyanenko, Dmitry 57
Er, Meng Joo 23
Eriguchi, Akiko 73, 81
Escolano, Carlos 51, 56
Esipova, Masha 47
Espeholt, Lasse 29, 35

Fairley, Richard E 134
Fan, Angela 35, 61, 126
Faruqui, Manaal 21
Farwell, David 97
Federico, Marcello 45, 53, 75, 78, 87
Federmann, Christian 2, 43, 54, 56, 68, 71, 77, 78, 112, 170
Fedus, William 69
Fei-Fei, Li 2, 69
Felice, Mariano 180, 183
Felix, Matthieu 126
Feng, Jiangtao 73
Feng, Minwei 36
Feng, Shi 69, 72
Feng, Xiaocheng 45
Feng, Yang 74, 97
Fergus, Rob 29, 80
Ferreira, Pedro 61
Finch, Andrew 45, 71, 76, 226
Firat, Orhan xxvi, 36–38, 44, 50, 54, 55, 59, 72, 78, 79, 82, 87, 95, 112, 126, 141, 145, 185, 186, 190, 195
Fiscus, J. G. 97
Fiscus, Jonathan 77
Fishel, Mark 43, 54, 56, 70, 71, 76–78, 112, 170
Fohr, Dominique 87
Fonollosa, José AR 51, 56
Forcada, Mikel L. 75
Fossum, Victoria 8, 94, 124, 125, 128, 131
Foster, George 36–38, 54, 72, 75, 76, 87, 112
Frank, Stella 80
Frasconi, Paolo 24, 35, 61
Fraser, Alexander 51, 53, 215, 217, 221, 229
Frederking, Robert 97
Freitag, Markus 43, 46, 50, 76, 80

French, Robert M. 76
Fügen, Christian 47
Fujita, Atsushi 56, 88

Gage, Philip 52
Gales, Mark JF 46, 105
Galley, Michel 46
Gangi, Mattia Antonino Di 75, 87
Gao, Jiameng 149
Gao, Jianfeng 29, 46, 75, 76, 174
Gao, Qin 2, 20, 37, 38, 42, 46, 52, 59, 60, 67, 79, 109, 118, 141, 151, 170
García-Martínez, Mercedes 82
Gardner, Matt 21
Garmendia Arratibel, Lierni 87
Gaspari, Federico 87
Gatti, Maira 23
Gauvain, Jean-Luc 8, 19
Ge, Qi 177, 179, 209
Ge, Tao 174, 181
Gehring, Jonas xviii, 34, 35, 37, 38
Geng, Xinwei 45
Georgakopoulou, Panayota 87
Germann, Ulrich 43, 77, 114, 116, 118
Ghader, Hamidreza 70, 215
Ghazvininejad, Marjan 75
Ghemawat, Sanjay 29, 124, 128, 148, 198
Ghoshal, Arnab 12
Gialama, Maria 87
Giles, C Lee 74
Gilroy, Sorcha 140
Gimpel, Kevin 46
Ginsburg, Boris 126
Girletti, Sabrina 87
Gitman, Igor 126
Glass, James 69, 70
Godard, Pierre 126
Goel, Vaibhava 105
Gokrani, Aman 76
Goldberg, Yoav 1, 19, 21, 71, 81, 203, 229
Golikov, Eugene 67
Gomez, Aidan N xviii, xxiv, 26–28, 35–38, 44, 64, 66, 71, 72, 114, 116–118, 124–126, 128, 141, 143, 148–150, 179, 195–199, 216, 222, 223, 227
Gong, Li 43, 45
Gong, Yifan 80
Gong, Yongen 174
Goodfellow, Ian 29, 32, 44, 68, 76, 93, 124, 126, 128, 148, 159
Gorman, Kyle 4, 201, 204, 205, 209, 233, 301
Goutte, Cyril 75, 76
Gouws, Stephan 114, 116, 117, 124–126, 128, 141, 148, 149, 179, 198, 222
Goyal, Anirudh Goyal Alias Parth 68
Goyal, Kartik 46
Grabska-Barwinska, Agnieszka 77
Graça, Miguel 76, 229
Graham, Yvette 43, 54, 56, 71, 77, 78, 112, 170
Grangier, David xviii, 34, 35, 37, 38, 50, 56, 61, 64, 66, 67, 70, 79, 126, 139
Grannes, Dean 97
Grave, Edouard 21
Graves, Alex 1, 29, 35, 74
Grefenstette, Edward 23, 29, 71, 74
Grissom II, Alvin 47, 69
Gross, Sam 126
Grundkiewicz, Roman 88, 174, 179–181
Gu, Jiatao 45–47, 71, 74
Guestrin, Carlos 2, 69

Gulcehre, Caglar xxvi, 19, 24, 25, 30, 31, 49, 51, 55, 61, 87, 93, 152, 157, 159, 185, 186, 190, 195
Guo, Junliang 74
Guo, Qipeng 72
Gupta, Suyog 79
Gurevych, Iryna 87
Guta, Andreas 71, 229
Gutmann, Michael 50
Gwinnup, Jeremy 76, 77

Ha, Thanh-Le 47, 57, 78, 88, 139
Haddow, Barry xxv, 42–45, 52, 54, 56, 61, 71, 76–79, 82, 102, 105, 112, 114–116, 118, 126, 132, 141, 149, 151, 159, 170, 178, 179, 185, 188, 189, 198, 222, 229
Hadiwinoto, Christian 179
Hadsell, Raia 77
Haffari, Gholamreza 43, 45, 56, 71, 81, 82, 88
Haffner, Patrick 32
Haghpanah, Yasaman 77
Hajishirzi, Hannaneh 82
Han, Song 79
Han, Xianpei 74
Hanazawa, Toshiyuki 32
Hannun, Awni 46
Hans, Krupakar 51
Hansen, Jonas Meinertz 51
Hansen, Lars Kai 42, 151
Hao, Jie 37
Harbecke, David 2, 69
Harley, Tim 74
Hashimoto, Kazuma 73, 81
Hasler, Eva 3, 4, 10, 50, 71, 80, 85–89, 97, 106, 112, 120, 121, 123, 124, 126, 128, 130, 138, 141, 150, 169, 170, 172, 173, 299, 300
Hassabis, Demis 77
Hassan, Hany 2, 56, 68
Hassibi, Babak 79, 152
Hayashibe, Yuta 179
Hazan, Elad 158
He, Di 45, 56, 74
He, He 47
He, Kaiming 37, 64
He, Tianyu 74
He, Wei 56, 59, 67, 68, 78, 87, 88, 141
He, Xiaodong 29, 75, 76
He, Xuanli 43
He, Zhongjun 47, 56, 59, 67, 68, 74, 80, 82, 87, 88, 141
Heafield, Kenneth 8, 43, 48, 56, 75, 79, 81, 93, 114, 116, 118, 128, 131, 177, 229
Hebb, Donald 155
Heess, Nicolas 29
Helcl, Jindřich 74, 126
Helle, Alexandre 51
Helmreich, Stephen 97
Henderson, Donnie 32
Henderson, James 82
Hermann, Karl Moritz 29, 74
Hethnawi, Mohammed 71
Hewitt, John 126
Hewlett, Daniel 71
Hieber, Felix 56, 126
Hildebrand, Almut Silja 76
Hinton, Geoffrey E. 21, 29, 32, 37, 61, 64–66, 80, 81, 116
Hitschler, Julian 80, 126
Hoang, Cong Duy Vu 71, 88
Hoang, Hieu 44, 79, 82, 87, 117, 126, 138, 220
Hoang, Vu Cong Duy 45, 56
Hochreiter, Sepp 24, 31, 35, 61, 74, 173

Hofmann, Thomas 51, 54
Hoshen, Yedid 77
Hovy, Eduard 2, 60, 67, 69, 87, 97
Howard, Andrew G 34
Howard, Richard E. 79, 152
Hu, Baotian 74
Hu, Junjie 77
Hu, Ke 2
Hu, Minghao 36
Hu, Qinghao 79
Hu, Wenpeng 88
Hu, Xiaoguang 74
Hu, Zhiting 72
Huang, Da 73
Huang, Eric H. xvii, 21, 22
Huang, Fei 70
Huang, Liang 47, 60, 141
Huang, Po-Sen 73
Huang, Po-Yao 80
Huang, Shudong 77
Huang, Shujian 43, 54, 56, 77, 81
Huang, Sitao 73
Huang, Xuedong 2, 56, 68
Huang, Zhen 36
Huang, Zhongqiang 19, 50
Hubbard, Wayne E 32
Huck, Matthias 43, 53, 54, 56, 77
Hughes, Macduff 36–38, 72, 76, 78, 112
Hula, Jan 21
Hussain, Aman 74
Hyvärinen, Aapo 50

Iglesias, Gonzalo 3, 10, 12, 16, 57, 70, 71, 93, 99, 100, 106, 110, 112, 121, 123, 126, 128, 132, 139, 141, 146, 150, 164, 300
Im, Jinbae 29
Imamura, Kenji 56
Ioffe, Sergey 37, 58, 64, 65, 116
Irving, Geoffrey 29, 124, 128, 148, 198
Isabelle, Pierre 87
Isahara, Hitoshi 109, 160, 222
Isard, Michael 29, 124, 128, 148, 198
Ishiwatari, Shonosuke 66, 73, 117
Ittycheriah, Abe 50, 60, 71
Iyyer, Mohit 21, 69, 74, 81

Jaakkola, Tommi 2, 69
Jackel, Lawrence D. 79, 152
Jain, Sarthak 215
Jaitly, Navdeep 1, 2, 29, 65, 67, 71, 204, 205, 209, 211
Jan, Niehues 78
Jauregi Unanue, Inigo 87
Jauvin, Christian 8, 19, 20, 192
Jean, Sebastien 82
Jégou, Hervé 77
Jelinek, Fredrick 8
Jernite, Yacine 51
Ji, Jianshu 174
Ji, Rongrong 45, 81
Jia, Ruoyu 181
Jia, Weijia 73
Jia, Yanfang 87
Jia, Ye 1
Jiang, Jing 23, 29, 36, 71
Jiang, Liyang 43, 45, 55, 170
Jimeno Yepes, Antonio 43, 54
Jin, Zhi 23
Johansen, Alexander Rosenberg 51
Johnson, Justin 2, 69
Johnson, Melvin 36–38, 72, 78, 112
Jones, Llion xviii, 26–28, 35–38, 44, 64, 66, 71, 72, 112, 114, 116–118, 124–126, 128,

141, 148–150, 179, 195–199, 216, 222, 223, 227
Joulin, Armand 74
Jouvet, Denis 87
Joy, David 77
Jun, Heewoo xxvi, 185, 186, 190
Junczys-Dowmunt, Marcin 2, 43, 44, 56, 68, 76, 81, 87, 88, 117, 126, 138, 170, 174, 179–181, 203
Jurafsky, Dan 2, 46, 68, 69, 174

Kahembwe, Emmanuel 72
Kaiser, Łukasz xviii, xxiv, 26–28, 35–38, 44, 45, 64, 66, 71, 72, 74, 78, 81, 112, 114, 116–118, 124–126, 128, 141, 143, 148–150, 179, 195–199, 216, 222, 223, 227
Kalaidin, Pavel 36
Kalchbrenner, Nal xvii, 1, 19, 22–24, 35, 114, 116, 117, 124–126, 128, 141, 148, 149, 179, 198, 222
Kalenichenko, Dmitry 34
Kalita, Jugal 72
Kang, Tae Gyoon 80
Kann, Katharina 79
Kaplan, Jared 66
Karafiát, Martin 186, 192
Karlen, Michael 21
Karpathy, Andrej 2, 69
Karpukhin, Vladimir 75
Kathirvalavakumar, T. 79
Katsuitho, Sudoh 78
Kavukcuoglu, Koray 1, 21, 29, 35, 50
Kay, Will 29
Kazawa, Hideto 77
Kazimi, M Bashir 61
Kell, Gregory 149
Keneshloo, Yaser 67
Kermorvant, Christopher 12
Keskar, Nitish Shirish 72
Keung, Phillip 78
Khadivi, Shahram 71, 75, 88
Khalil, Talaat 20
Khalilov, Maxim 20
Khayrallah, Huda 75–77, 87, 88
Khudanpur, Sanjeev 12, 186, 192
Kiefer, Jack 114, 148
Kiela, Douwe 23
Kikuchi, Yuta 61
Kilgour, Kevin 47
Kim, Been 2, 69
Kim, Ho-Gyeong 80
Kim, Jungi 20, 76
Kim, Yoon 23, 46, 51, 80, 81, 126, 174
Kim, Young-Kil 79
Kim, Yunsu 76
Kindermans, Pieter-Jan 66, 117
Kingma, Diederik P 116
Kirkpatrick, James 77
Kiros, Jamie R 29, 37, 45, 64, 71, 116
Kitsuregawa, Masaru 73
Klauschen, Frederick 69
Klein, Dan 50
Klein, Guillaume 20, 126
Knibbe, Maxime 12
Knight, Kevin 9, 44, 52, 77, 79, 86, 97, 140
Knowles, Rebecca 58, 71, 75, 77, 87, 214
Kobus, Catherine 76
Kocisky, Tomas 29, 71
Koehn, Philipp 7, 8, 10, 11, 15, 20, 39, 41, 43, 46, 48, 54–56, 58, 69–71, 75–78, 81, 82, 87, 88, 93, 112, 128, 131, 170, 177, 179, 193, 203, 209, 214, 220, 229
Koichiro, Yoshino 78

Kolss, Muntsin 47
Komachi, Mamoru 179
Koncel-Kedziorski, Rik 82
Kong, Lingpeng 23, 73
Kong, Xiang 60, 67, 87
Koo, Terry 81
Kovalev, Fedor 20
Krause, Ben 72
Kretov, Maksim 67
Kreutzer, Julia 54
Krikun, Maxim 2, 20, 37, 38, 42, 46, 52, 59, 60, 67, 76, 78, 79, 109, 118, 141, 151, 170
Krishna, Kalpesh 74, 81
Krislauks, Rihards 79
Krizhevsky, Alex 37, 65, 116
Krüger, Bastian 47
Kruszewski, Germán 23
Kuang, Shaohui 76, 82
Kuchaiev, Oleksii 126
Kudlur, Manjunath 29, 124, 128, 148, 198
Kudo, Taku 52, 66
Kuhn, Roland 75, 76
Kuksa, Pavel 21
Kumar, Aviral 58, 70
Kumar, Gaurav 76, 88
Kumar, Shankar 12, 105, 106, 108, 113
Kumaran, Dharshan 77
Kunchukuttan, Anoop 52, 77, 79
Kuncoro, Adhiguna 81
Kurach, Karol 74
Kurimo, Mikko 105
Kurohashi, Sadao 42, 43, 76, 109, 151, 160, 222

Labaka, Gorka 77
Lacoste, Alexandre 71
Ladhak, Faisal 78
Lafferty, John D. 8
Lai, Jianhuang 67–69
Lakew, Surafel Melaku 78
Lala, Chiraag 80
Lamar, Thomas 19, 50
Lamb, Alex M 68
Lamblin, Pascal 29, 93, 124, 126, 128, 148, 159
Lample, Guillaume 23, 77, 78
Lang, Kevin J 32
Langlois, David 87
Lapata, Mirella 29, 35, 82
Larkin, Samuel 76
Larochelle, Hugo 29, 69, 74
Läubli, Samuel 2, 82, 126, 195
Lauly, Stanislas 82
Lavrukhin, Vitaly 126
Lawson, Dieterich 71
Le, Hai-Son 19
Le Nguyen, Minh 45
Le, Quoc V xvii, 2, 20, 21, 24, 25, 29, 30, 37, 38, 42, 46, 49, 52, 57, 59, 60, 66, 67, 72, 76, 78, 79, 92, 94, 109, 111, 117, 118, 141, 151, 170
Lecorvé, Gwénolé 46
LeCun, Yann 32, 79, 80, 152
Lee, Chen-Yu 29
Lee, Hodong 47
Lee, Hoshik 80
Lee, Jason 51, 54, 74
Lee, Jihyun 47, 80
Lee, Kenton 21, 72
Lee, Min-Joong 80
Lee, Stefan 47
Lee, Yee-Chun 74
Leng, Cong 79

Levenberg, Josh 29, 124, 128, 148, 198
Levenshtein, Vladimir I 97
Levin, Pavel 20
Levy, Omer 75
Lewis II, Philip M 15
Lewis, William D 2, 47, 56, 68, 75
L'Hostis, Gurvan 50
Li, Aodong 45, 79
Li, Fuxue 49
Li, Ge 23
Li, Hang 29, 56, 60, 74, 87, 88
Li, Hanji 2
Li, Jinyu 80
Li, Jiwei 2, 46, 68, 69
Li, Ke 69
Li, Mu 2, 37, 45, 56, 68, 69, 72, 73, 75, 77, 79, 81
Li, Muyu 73
Li, Muze 43, 45, 55, 170
Li, Peng 22
Li, Shaotong 49
Li, Victor OK 74
Li, Xian 69
Li, Xiaoqing 49
Li, Xing 47
Li, Ya 74
Li, Yachao 73
Li, Yancui 49
Li, Yanyang 60
Li, Yinqiao 60
Liang, Chen 79
Liang, Shangsong 23, 36
Libovický, Jindřich 126
Lillicrap, Timothy 23
Lin, Huei-Chi xxvi, 55, 87, 185, 186, 190, 195
Lin, Jimmy 48
Lin, Junyang 37, 45, 73
Lin, Lei 29, 36
Lin, Qian 60, 87
Lin, Zhouhan 36
Ling, Wang 23, 51, 229
Linzen, Tal 69
Lipton, Zachary C. 2, 69
Liu, Chao-Hong 87
Liu, Dan 74
Liu, Frederick 80
Liu, Guiquan 56
Liu, Hairong 47
Liu, Jingming 181
Liu, Junhua 74
Liu, Kaibo 47
Liu, Lemao 45, 71, 76, 81, 226
Liu, Nelson F 52
Liu, Pengfei 72
Liu, Peter J 57, 71
Liu, Qi 45
Liu, Qiuhui 126
Liu, Qun 43, 54, 56, 74, 77, 80, 82, 97
Liu, Shujie 45, 56, 68, 69, 72, 73, 75, 77, 78, 88
Liu, Tianyu 23, 36
Liu, Tie-Yan 45, 56, 67–69, 74
Liu, Ting 45
Liu, Xiaohua 56, 60, 87
Liu, Xinyu 74
Liu, Xunying 46
Liu, Yang 2, 22, 29, 36, 45, 56, 60, 67–69, 75, 78, 81, 82, 87, 97, 126, 214, 215
Liu, Yijia 170
Liu, Yizhu 61
Liu, Yuchen 44, 80
Livni, Roi 64

Logacheva, Varvara 43, 54, 56, 70, 77
Lohar, Pintu 87
Long, Guodong 23, 29, 36, 71
Long, Zi 88
Lopez, Adam 220
Louradour, Jérôme 76
Lu, Hanqing 45
Lu, Jianfeng 72
Lu, Yaojie 74
Lu, Yichao 78
Lu, Zhengdong 29, 60, 74, 87, 88
Lu, Zhiyun 80
Luan, David 72
Luan, Huanbo 2, 69, 82, 126, 214, 215
Luan, Yi 82
Luitjens, Justin 12
Luo, Weihua 82
Luo, Zhiyi 61
Luong, Minh-Thang 27, 49, 51, 71, 72, 76, 78, 79, 94, 111, 116, 188
Lynn, Teresa 87

Ma, Chunpeng 81
Ma, Mingbo 47, 60, 141
Ma, Shuai 77, 78, 88
Ma, Shuming 23, 36, 37, 45
Ma, Wei-Ying 56
Ma, Xutai 69
Macháček, Dominik 53
Macherey, Klaus 2, 20, 37, 38, 42, 46, 52, 59, 60, 67, 79, 109, 118, 141, 151, 170
Macherey, Wolfgang 2, 20, 36–38, 42, 46, 52, 54, 59, 60, 67, 72, 79, 94, 99, 105, 106, 108, 109, 112, 113, 118, 141, 147, 151, 170
Macketanz, Vivien 2, 69, 88, 112, 120, 121
Mahata, Sainik Kumar 87
Maillard, Jean 81
Makhoul, John 19, 50
Malaviya, Chaitanya 61
Maletti, Andreas 140
Malik, Jitendra M 134, 136, 137
Malinowski, Mateusz 23
Malykh, Valentin 36
Mandal, Soumil 87
Manning, Christopher D 21, 76, 79, 81
Mansimov, Elman 74
Mao, Jiayuan 73
Marcheggiani, Diego 81, 82
Marcinkiewicz, Mary Ann 172
Marcus, Mitchell P 172
Marg, Lena 20
Marie, Benjamin 88
Martins, André F. T. 61, 70, 82
Maruf, Sameen 82
Marvin, Rebecca 76
Matsumoto, Yuji 179
Matthews, Austin 126, 229
Matusov, Evgeny 71, 88
May, Jonathan 52, 77, 140
McCandlish, Sam 66
McCann, Bryan 21
McCarthy, Arya D. 76
McCoy, R Thomas 21
McGraw, Lan 80
McNamee, Paul 76
Mediani, Mohammed 47
Medina, Julian Richard 72
Mehri, Shikib 45
Melamed, I. Dan 218
Melis, Gábor 81
Mella, Odile 87
Memisevic, Roland xxiii, 49, 50, 55, 59, 92–95, 141, 145

Men, Rui 23
Menacer, Mohamed Amine 87
Meng, Fandong 72, 75, 88
Mercer, Robert L. 8–10, 55, 71, 88, 220
Mesnil, Grégoire 46
Mi, Haitao 50, 60, 71, 97
Miao, Guoyi 49
Miceli Barone, Antonio Valerio 43, 56, 77, 114, 116, 118, 126
Michel, Paul 69, 75
Micikevicius, Paulius 126
Miculicich, Lesly 37, 82
Mieno, Takashi 47
Mikolov, Tomáš 186, 192
Miks, Toms 53
Milan, Kieran 77
Milton, RS 51
Mino, Hideya 27
Mirza, Mehdi 32, 68, 76, 93, 159
Mitchell, Tom 76
Mitsuhashi, Tomoharu 88
Mizumoto, Tomoya 179
Mnih, Andriy 50
Mnih, Volodymyr 29
Mohri, Mehryar 12–14, 91, 98, 100, 109, 124, 135, 146, 149, 175, 178
Mokry, Jozef 126
Möller, Sebastian 2, 69
Monga, Rajat 29, 124, 128, 148, 198
Monroe, Will 68
Montavon, Grégoire 2, 69
Monz, Christof 43, 54, 56, 70, 71, 75–78, 112, 170, 215
Moore, Sherry 29, 124, 128, 148, 198
Moorkens, Joss 87
Morgan, John 47
Morin, Fréderic 8, 19
Morishita, Makoto 66, 85, 88, 94, 160
Mosca, Abigail 69
Motlicek, Petr 46
Mou, Lili 23
Moussallem, Diego 82
Moysset, Bastien 12
Müller, Klaus-Robert 2, 69
Müller, Markus 47
Müller, Mathias 36, 72, 82
Munkhdalai, Tsendsuren 74
Murray, Derek G. 29, 124, 128, 148, 198
Murray, Iain 72
Murray, Kenton xviii, 58, 59, 141
Murthy, Rudra 77

Na, Hwidong 47, 80
Nadejde, Maria 81, 126, 203
Nagata, Masaaki 179
Nakagawa, Tetsuji 76, 77
Nakajima, Kaisuke 52
Nakamura, Satoshi 47, 66, 79, 85, 88, 94, 108, 109, 160
Nakazawa, Toshiaki 20, 42, 43, 109, 151, 160, 222
Nakov, Preslav 54
Nallapati, Ramesh 49
Napoles, Courtney 179
Narasimhan, Karthik 21, 72
Negri, Matteo 43, 53, 54, 56, 77
Neishi, Masato 66, 117
Neubig, Graham xxv, 20, 41–43, 46, 47, 52, 54, 56, 61, 66, 69, 71, 75–77, 79, 81, 85, 88, 94, 108–110, 126, 149, 151, 160
Neumann, Mark 21
Névéol, Aurélie 222
Neves, Mariana 43, 54, 222

Ney, Hermann 10, 11, 70, 71, 74, 76, 79, 88, 222, 223, 225, 229
Ng, Andrew Y 174
Ng, Axel H. 4, 201, 204, 205, 209, 233, 301
Ng, Hwee Tou 174, 179
Ng, Nathan 126
Ngomo, Axel-Cyrille Ngonga 82
Nguyen, Thai Son 47
Nguyen, Toan Q. 50, 77
Niculescu-Mizil, Alexandru 80
Niehues, Jan 47, 52, 54, 57, 78, 81, 88, 139
Nielsen, Henrik 29
Nirenburg, Sergei 97
Nishimura, Yuta 79
Niu, Xing 56, 75
Nivre, Joakim 70, 72
Norouzi, Mohammad 2, 20, 37, 38, 42, 43, 46, 52, 59, 60, 67, 79, 109, 118, 141, 151, 170
Nortonsmith, Avery 69
Norvig, Peter 134, 136, 137

Obeid, Elias Khazen 51
Och, Franz J 10, 11, 59, 94, 99, 105, 106, 108, 109, 113, 147, 185, 222, 223, 225
Och, Franz Josef 79
Oda, Yusuke 66, 109
Oh, Jean 80
Ojha, Atul Kr 87
Okazaki, Naoaki 61
Okumura, Manabu 61
Osindero, Simon 29
Östling, Robert 45, 75, 77
Ott, Myle 56, 64, 66, 67, 70, 78, 79, 126, 139
Ozair, Sherjil 68

Padmanabhan, Sarguna 126
Pal, Christopher 29
Pal, Santanu 229
Paliwal, Kuldip K 25, 31
Palm, Rasmus 23
Pamar, Niki 45
Pan, Shirui 23, 29, 36
Papineni, Kishore 57, 67, 70
Pappagari, Raghavendra 21
Pappas, Nikolaos 37, 82
Paquet, Ulrich 23
Parikh, Ankur 36
Park, Youngki 47
Parmar, Niki xviii, 26–28, 35–38, 44, 64, 66, 71, 72, 112, 114, 116–118, 124–126, 128, 141, 148–150, 179, 195–199, 216, 222, 223, 227
Pascanu, Razvan 23, 29, 64, 77, 93, 124, 126, 128, 148, 159
Pascual, Damián 215
Passban, Peyman 56, 191
Patel, Roma 21
Patterson, Geneviève 69
Paulik, Matthias 47
Paulus, Romain 36
Pavlick, Ellie 21
Peng, Furong 72
Peng, Xiaochang 4, 201, 204, 205, 209, 233, 301
Peng, Yuxing 36
Pennington, Jeffrey xvii, 21, 22
Pereira, Fernando 12
Pereyra, Gabriel 66
Peter, Jan-Thorsten 71, 229
Peters, Matthew 21
Peterson, Kay 77
Petrov, Slav 81
Petrushkov, Pavel 88

Pham, Hieu 27, 56, 66, 72, 81, 188
Piccardi, Massimo 87
Pineau, Joelle 69
Pinnis, Mārcis 53
Pino, Juan 69
Platanios, Emmanouil Antonios 76
Poczos, Barnabas 76
Pollack, Jordan B. 21
Polosukhin, Illia xviii, 26–28, 35–38, 44, 64, 66, 71, 72, 114, 116–118, 141, 148, 150, 195–197, 199, 216, 222, 223, 227
Poncelas, Alberto 56, 87, 191
Ponkratov, Anton 57
Pool, Jeff 79
Popat, Ashok C. 185
Popel, Martin 43, 44, 48, 54, 66, 117, 223
Popescu-Belis, Andrei 20, 37
Post, Matt 43, 54, 56, 76, 77, 88, 126, 174, 179, 188, 203
Pouget-Abadie, Jean 26, 68
Pouzyrevsky, Ivan 8, 48, 93, 128, 131, 177
Povey, Daniel 12
Powell, Michael JD 113, 148
Power, Russell 21
Prabhavalkar, Rohit 80
Pramanik, Subhojeet 74
Pratama, Mahardhika 23
Pryzant, Reid 76
Puduppully, Ratish 170
Pust, Michael 52

Qi, Ye 126
Qiang, Wang 49
Qiao, Kan 73
Qin, Bing 45, 170
Qin, Tao 45, 56, 67–69, 74
Qin, Yue 45, 60, 87
Qiu, Xipeng 36, 72
Quan, Du 49
Quan, John 77
Quinn, Jerry 79
Quirk, Chris 75

Rabinowitz, Neil 77
Radford, Alec 21, 72
Rafea, Ahmed 45
Raffel, Colin 71
Rahimi, Ali 2
Ram, Dhananjay 37, 82
Ramachandran, Prajit 57
Ramakrishnan, Naren 67
Ramalho, Tiago 74, 77
Ranzato, Marc'Aurelio 64, 67, 70, 77, 78, 80, 139
Raposo, David 23
Rarrick, Spencer 75
Rebollo, Anabel 20
Reddy, Chandan K 67
Reddy, Siva 81, 203
Rei, Marek xx, 124, 182, 184, 185, 301
Reichhold, Jane xvii, 29
Ren, Shaoqing 37, 64
Ren, Shuo 77, 78, 88
Ren, Xuancheng 37, 45, 73
Renals, Steve 72, 76
Resnik, Philip 75
Reynolds, Malcolm 74
Riad, Rachid 126
Ribeiro, Marco 2, 69
Riccardi, Giuseppe 12
Richardson, John 52
Richter, Oliver 215
Riess, Simon 53
Riezler, Stefan 80

Rifai, Salah 46
Rikters, Matīss 70, 73
Riley, Michael D 12, 14, 16, 91, 100, 109, 124, 128, 146, 149, 175, 177, 178, 213
Rios, Annette 36, 72, 82
Ritter, Alan 68
Roark, Brian 4, 177, 201, 204, 205, 209, 213, 233, 301
Robinson, Shannon 69
Robinson, Tony 177, 179, 209
Rodriguez, Pedro 69
Rodriguez, Tim 175
Rokach, Lior 44
Roossin, Paul S. 8
Rosendahl, Jan 76
Rossenbach, Nick 76, 229
Roth, Dan 21
Roukos, Salim 57, 67, 70
Roy, Aurko 45
Rubino, Raphael 43, 54, 56, 77
Ruiz, Nicholas 75, 87
Rumelhart, David E. 21, 61
Runge, Andrew 52, 54
Rush, Alexander M 81, 126, 174
Russell, Stuart Jonathan 134, 136, 137
Rusu, Andrei A. 77
Rykachevskiy, Anton 57

Sachan, Devendra 126
Sahbi, Hichem 105
Sainath, Tara N 80
Sajjad, Hassan 42, 43, 54, 69, 70, 76, 151, 229
Sakaguchi, Keisuke 174, 179
Sakti, Sakriani 47, 109
Sakuma, Jin 66, 117
Salakhutdinov, Ruslan 29, 37, 65, 71, 72, 116
Salamon, Peter 42, 151
Salesky, Elizabeth 52, 54
Salimans, Time 21, 72
Samek, Wojciech 2, 69
Sánchez-Cartagena, Víctor M. 87
Sankaran, Baskaran 44, 50, 60, 78–80, 128
Santorini, Beatrice 172
Santoro, Adam 23
Santos, Cicero Nogueira dos 36
Sarawagi, Sunita 26, 57, 58, 70
Sarkar, Anoop 128
Sasano, Ryohei 61
Satheesh, Sanjeev xxvi, 185, 186, 190
Satta, Giorgio 218
Saunders, Danielle xxvi, 3, 4, 57, 77, 81, 85, 86, 117, 120, 121, 123, 124, 126, 132, 138, 139, 141, 149, 150, 169, 170, 179, 195, 198, 199, 201–203, 223, 299–301
Saxena, Karan 87
Scarton, Carolina 43, 54
Schalkwyk, Johan 12, 14, 109, 124, 146, 149, 175
Schamoni, Shigehiko 80
Scherrer, Yves 82
Schmaltz, Allen xxv, 170–174
Schmid, Helmut 215, 217, 221, 229
Schmidhuber, Jürgen 24, 35, 61
Schmidt, Tanja 20
Schnober, Carsten 87
Schuster, Mike 2, 20, 25, 31, 36–38, 42, 46, 52, 59, 60, 67, 72, 78, 79, 109, 112, 118, 141, 151, 170, 177, 179, 209
Schütze, Hinrich 29, 79, 229
Schwartz, Richard 19, 50
Schwarzenberg, Robert 2, 69

Schwenk, Holger xxvi, 8, 19, 23–25, 30, 31, 55, 56, 61, 87, 92, 93, 152, 157, 159, 185, 186, 190, 195
Sculley, D 2
SDL 2
Seal, Matthew 175
Sebastian, Stüker 78
See, Abigail 79
Selvaraju, Ramprasath R 47
Senécal, Jean-Sébastien 8, 19
Senellart, Jean 20, 76, 80, 126
Senior, Andrew 1
Sennrich, Rico xxv, 2, 36, 42–45, 52, 54, 56, 61, 70, 72, 76, 77, 79, 81, 82, 87, 88, 102, 105, 114–118, 126, 132, 141, 149, 151, 159, 178, 179, 185, 188, 189, 195, 198, 203, 222
Sepassi, Ryan 114, 116, 117, 124–126, 128, 141, 148, 149, 179, 198, 222
Serdyuk, Dmitriy 93, 124, 126, 128, 134, 138, 148, 149, 159
Serdyukov, Pavel 82
Serrano, Sofia 215
Shah, Harshil 74
Shakhnarovich, Gregory 46
Shakibi, Babak 80
Shalev-Shwartz, Shai 64
Shamir, Ohad 64
Shang, Lifeng 29, 56
Shao, Yunfan 72
Shaw, Peter 37, 72, 116
Shazeer, Noam xviii, 26–28, 35–38, 44, 45, 64, 66, 67, 71, 72, 112, 114, 116–118, 124–126, 128, 141, 148–150, 179, 195–199, 216, 222, 223, 227
Shen, Kewei 181
Shen, Shiqi 67, 68, 126
Shen, Tao 23, 29, 36, 71
Shen, Yikang 81
Sheng, Quan Z 69
Shi, Lin 43, 45, 55, 170
Shi, Shuming 60, 67, 82, 87
Shi, Tian 67
Shi, Tianlin 68
Shi, Xiaodong 81
Shiang, Sz-Rung 80
Shieber, Stuart xxv, 170–174
Shikano, Kiyohiro 32
Shin, Jong-Hun 79
Shlens, Jon 58, 65, 116
Shrivastava, Manish 170
Shterionov, Dimitar 56, 191
Shuyo, Nakatani 92, 114
Siahbani, Maryam 128
Siegelmann, Hava T 73
Sigal, Leonid 45
Sim, Khe Chai 105
Simaan, Khalil 81
Simonyan, Karen 1, 35
Sindhwani, Vikas 80
Singer, Yoram 158
Singh, Sameer 2, 69
Singh, Sumeet 75
Skorokhodov, Ivan 57
Skut, Wojciech 12, 14, 109, 124, 146, 149, 175
Slotin, Sergey 57
Smaïli, Kamel 87
Smaragdis, Paris 80, 158
Smith, Noah A. 75, 81, 215, 229
Smith, Samuel L 66, 117
Smola, Alex 29, 37, 72, 79
Snoek, Jasper 2

So, David R 79
Soboroff, Ian 77
Socher, Richard xvii, 21, 22, 36, 72, 74, 81
Sokolov, Artem 54, 126
Solla, Sara A. 79, 152
Sønderby, Casper Kaae 29, 51
Sønderby, Søren Kaae 29
Song, Inchul 47
Song, Kaitao 72
Sontag, David 51
Sontag, Eduardo D 73
Sordoni, Alessandro 81
Sorensen, Jeffrey 177, 213
Sosoni, Vilelmini 87
Sountsov, Pavel 26, 57
Specia, Lucia 43, 54, 56, 70, 77, 80
Sperber, Matthias 47, 81, 126
Sproat, Richard 2, 4, 29, 177, 201, 204, 205, 209, 211, 213, 233, 301
Srinivas, Suraj 80, 152, 155–158, 161
Sriram, Anuroop xxvi, 185, 186, 190
Srivastava, Nitish 37, 65, 116
Stahlberg, Felix xx, xxvi, 3, 4, 12, 43–45, 50, 57, 64, 66, 77, 81, 85–89, 97, 117, 120, 121, 123, 124, 126, 128, 130, 132, 138, 139, 141, 149–151, 169, 170, 172, 173, 179, 182–185, 195, 197–199, 201–205, 209, 223, 233, 299–301
Stearns, Richard Edwin 15
Steiner, Benoit 29, 124, 128, 148, 198
Stenetorp, Pontus 229
Stern, Mitchell 45
Stewart, Craig 75
Stolcke, Andreas 128, 131, 173
Stork, David G. 79, 152
Stoyanov, Veselin 4, 169, 185, 300
Stretcu, Otilia 76
Stüker, Sebastian 47
Su, Jinsong 37, 45, 60, 73, 74, 81, 87
Su, Qi 37, 45, 73
Sudoh, Katsuhito 66, 79
Sukhbaatar, Sainbayar 29
Suleyman, Mustafa 29, 74
Sumita, Eiichiro 27, 45, 46, 56, 71, 76, 81, 109, 126, 160, 222, 226
Sumita, Eiichro 88, 108
Sun, Chengjie 29, 36
Sun, Guo-Zheng 74
Sun, Jason 78
Sun, Jian 37, 64
Sun, Maosong 2, 22, 56, 67–69, 78, 82, 126, 214, 215
Sun, Qing 47
Sun, Xu 37, 45, 73
Susanto, Raymond Hendy 179
Sutskever, Ilya xvii, 20, 21, 24, 25, 30, 37, 42, 49, 65, 72, 74, 78, 81, 92, 94, 111, 116, 128, 131, 151, 173
Swersky, Kevin 71
Swietojanski, Pawel 76
Synnaeve, Gabriel 46
Szegedy, Christian 37, 58, 64, 65, 116
Szlam, Arthur 29

Täckström, Oscar 36
Taghipour, Kaveh 75, 174
Tai, Kai Sheng 81
Tai, Terry 177, 213
Takamura, Hiroya 61
Takase, Sho 61
Tambellini, William 3, 106, 112, 121, 150
Tamchyna, Aleš 51
Tamura, Akihiro 81

Tan, Shawn 81
Tan, Xu 74
Tan, Zhixing 81
Tang, Gongbo 36, 70, 72
Tang, Yaohua 88
Tars, Sander 76
Team, OpenAI Dota 66
Teh, Yee Whye 50
Tesauro, Gerald 74
Tetreault, Joel 179
Thayer, Ignacio 94, 99, 109, 113, 147
Thompson, Brian 76, 77
Thorat, Nikhil 78
Tian, Fei 67–69, 74
Tiedemann, Jörg 45, 75, 77, 82
Tillmann, Christoph 88
Tinsley, John 87
Titov, Ivan 74, 81, 82
Toda, Tomoki 47, 109
Tohda, Satoshi 66, 117
Tokunaga, Takenobu 27
Tomalin, Marcus 4, 124, 169, 170, 172, 173, 300
Tomasello, Laura 2
Tomczak, Marcin 124, 130, 149
Tong, Audrey 77
Tong, Xiao 49
Torabi, Atousa 29
Toral, Antonio 2, 87
Torregrosa, Daniel 79
Toutanova, Kristina 21, 72, 174
Toyoda, Masashi 66, 117
Tran, John 79
Tran, Ke 71
Trancoso, Isabel 51
Trischler, Adam 51
Tromble, Roy 105, 106, 108, 113
Truong, Steven 174
Tsuruoka, Yoshimasa 73, 81
Tu, Zhaopeng 37, 56, 60, 67, 72, 75, 82, 87, 88
Tucker, George 66, 71
Tucker, Paul 29, 124, 128, 148, 198
Turchi, Marco 43, 53, 54, 56, 77

Uchimoto, Kiyotaka 109, 160, 222
Ueffing, Nicola 70
Ullman, Jeffrey D. 15
Upadhyay, Shyam 21
Uszkoreit, Hans 88, 112, 120, 121
Uszkoreit, Jakob xviii, 26–28, 35–38, 44, 45, 64, 66, 71, 72, 94, 99, 109, 112–114, 116–118, 124–126, 128, 141, 147–150, 179, 195–199, 216, 222, 223, 227
Utiyama, Masao 27, 45, 46, 71, 76, 81, 88, 108, 109, 126, 160, 222, 226
Utsuro, Takehito 88

Vaibhav, Vaibhav 75
van den Oord, Aäron 1, 35
van der Wees, Marlies 75, 76
van Durme, Benjamin 21, 174
van Genabith, Josef 229
van Merriënboer, Bart 93, 124, 126, 128, 134, 138, 148, 149, 159
Vanhoucke, Vincent 58, 65, 116
Vasudevan, Vijay 29, 124, 128, 148, 198
Vaswani, Ashish xviii, 8, 26–28, 35–38, 44, 45, 64, 66, 71, 72, 94, 112, 114, 116–118, 124–126, 128, 131, 141, 148–150, 179, 195–199, 216, 222, 223, 227
Veness, Joel 77
Verspoor, Karin 43, 54

Vidra, Jonáš 53
Viégas, Fernanda 78
Vijayakumar, Ashwin K 47
Vilar, David 76, 126
Vincent, Pascal 8, 19, 20, 59, 74, 141, 145, 192
Vinyals, Oriol xvii, 1, 20, 24, 25, 29, 30, 42, 49, 67, 78, 80, 81, 92, 94, 111, 128, 131, 151, 173
Virpioja, Sami 105
Vogel, Stephan 12, 42, 43, 76, 88, 151, 229
Voita, Elena 82
Volk, Martin 2, 82, 195
Volkart, Lise 87
Vu, Tu 74
Vural, Fatos T. Yarman 78
Vyas, Yogarshi 75
Vymolova, Ekaterina 71, 88

Waibel, Alex 32, 47, 57, 76, 78, 81, 88, 139
Waite, Aurelien 3, 50, 85–89, 128, 130, 164, 299, 300
Wallace, Byron C. 215
Wallace, Eric 69
Wang, Alex 21
Wang, Boli 81
Wang, Changhan 45
Wang, Chenguang 37, 72, 79
Wang, Chong 73
Wang, Chunqi 74
Wang, Di 72
Wang, Dong 45, 74, 79
Wang, Feng 51, 68, 69
Wang, Haifeng 47, 59, 78, 80, 82, 87, 88, 141
Wang, Hongji 45
Wang, Houfeng 23, 36
Wang, Jun 68
Wang, Liang 181
Wang, Liming 126
Wang, Liwei 45, 56
Wang, Longyue 37, 82
Wang, Mingxuan 43, 45, 74
Wang, Ning 23
Wang, Qiang 60
Wang, Qinlong 174
Wang, Rui 46, 76, 81
Wang, Sen 36, 71
Wang, Wei 76, 86
Wang, Weijun 34
Wang, William Yang 78
Wang, Xiangling 87
Wang, Xiaolin 126
Wang, Xiaolong 29, 36
Wang, Xin 78
Wang, Xing 37, 88
Wang, Xinyi 56, 66, 81, 126
Wang, Yanfeng 43, 45, 55, 170
Wang, Yijun 56
Wang, Yiming 12
Wang, Yining 44
Wang, Yiren 74
Wang, Yongqiang 46
Wang, Yuguang 43, 45, 55, 170
Wang, Yuhang 79
Wang, Zhiguo 50, 60, 71
Wang, Zhiwei 149
Ward, Todd 57, 67, 70
Warde-Farley, David 29, 68, 93, 124, 126, 128, 134, 138, 148, 149, 159
Warden, Pete 29, 124, 128, 148, 198
Watanabe, Taro 76
Wattenberg, Martin 78
Wattenhofer, Roger 215

Watts, Nathan 69
Way, Andy 2, 56, 82, 87, 88, 191
Wayne, Greg 74
Wei, Furu 36, 174, 181
Weiss, Ron J. 1, 71
Welleck, Sean 45
Weller-Di Marco, Marion 51
Wellington, Benjamin 218
Wenniger, Gideon Maillette de Buy 56, 191
Werlen, Lesly Miculicich 37
Weston, Jason 21, 29, 76
Weyand, Tobias 34
Wicke, Martin 29, 124, 128, 148, 198
Wieting, John 23
Williams, Adina 81
Williams, Philip 43, 114, 116, 118, 203
Williams, Ronald J 74
Wiltschko, Alex 2
Winther, Ole 23, 29, 51
Wiseman, Sam 46
Wojna, Zbigniew 58, 65, 116
Wolf, Lior 77
Wong, Derek F. 81
Wong, Jeremy H.M. 80
Woodland, Philip C 46, 105
Woods, William A 16
Wu, Dekai 22, 74
Wu, Felix 35
Wu, Haiyang 72
Wu, Hua 56, 59, 67, 68, 74, 78, 80, 82, 87, 88, 141
Wu, Jeffrey 72
Wu, Jiawei 78
Wu, Jiaxiang 79
Wu, Lijun 67–69
Wu, Shan 74
Wu, Shuangzhi 45, 81
Wu, Siew Mei 179
Wu, Wei 23, 36
Wu, Yingting 52
Wu, Yonghui 2, 20, 36–38, 42, 46, 52, 59, 60, 67, 72, 78, 79, 109, 112, 118, 141, 151, 170
Wu, Yuanbin 179
Wuebker, Joern 71

Xia, Patrick 21
Xia, Yingce 45, 56, 68, 69, 74
Xiang, Bing 36
Xiao, Da 76
Xiao, Tong 60, 81
Xie, Jun 43, 45
Xie, Ziang 174
Xiong, Caiming 21, 36, 74
Xiong, Deyi 73, 74, 76, 81, 82, 88
Xiong, Hao 47, 74, 80, 82
Xu, Bing 68
Xu, Bo 51, 68, 69
Xu, Changming 60
Xu, Hainan 12
Xu, Hongfei 126
Xu, Jia 75
Xu, Jinan 49
Xu, Jingfang 82
Xu, Kelvin xxvi, 29, 55, 71, 87, 185, 186, 190, 195
Xu, Linli 74
Xu, Peng 185
Xu, Tan 72
Xu, Wei 56, 78
Xu, Xueying 76
Xu, Yan 23
Xue, Jian 80
Xue, Xiangyang 72

Yaguchi, Manabu 109, 160, 222
Yamamoto, Mikio 88
Yan, Rui 23
Yang, Baosong 37, 81
Yang, Hongtao 43, 45, 55, 170
Yang, Jiajun 43, 45, 55, 170
Yang, Jing 60, 87
Yang, Kathy 20
Yang, Nan 81
Yang, Qian 78
Yang, Yilin 60, 141
Yang, Yiming 72
Yang, Yuekui 72
Yang, Zhen 45, 51, 68, 69
Yang, Zhilin 72
Yang, Zichao 29, 72
Yannakoudakis, Helen xx, 124, 182, 184, 185, 301
Yao, Jingtao 73
Yao, Kaisheng 71, 88
Yao, Li 29
Yarats, Denis xviii, 34, 35, 37, 38
Yarman Vural, Fatos T. 44, 78, 79
Yepes, Antonio Jimeno 222
Yilmaz, Emine 23, 36
Yin, Pengcheng 81
Ying, Chris 66, 117
Yogatama, Dani 81
Yoshinaga, Naoki 66, 73, 117
Yoshino, Koichiro 66
Yu, Dianhai 78
Yu, Hong 74
Yu, Lantao 68
Yu, Lei 23, 71, 81
Yu, Mo 36
Yu, Nenghai 56, 68
Yu, Philip LH 88
Yu, Yong 68
Yu, Yuan 29, 124, 128, 148, 198
Yuan, Zheng xx, 29, 124, 174, 182, 184, 185, 301
Yuret, Deniz 77
Yvon, François 19, 56

Zamora-Martinez, Francisco 19
Zampieri, Marcos 43, 54, 229
Zare Borzeshi, Ehsan 87
Zaremba, Wojciech 49, 64, 67, 80, 94, 111, 128, 131, 173
Zaremoodi, Poorya 81
Zbib, Rabih 19, 50
Zeiler, Matthew D 61, 159
Zemel, Rich 29, 71
Zen, Heiga 1
Zenkel, Thomas 71
Zettlemoyer, Luke 21
Zhai, ChengXiang 74
Zhai, Feifei 82
Zhai, Junjie 72, 75
Zhang, Andi 74
Zhang, Biao 60, 73, 74, 87
Zhang, Chengqi 23, 29, 36, 71
Zhang, Chuanqiang 47
Zhang, Dakun 76, 80
Zhang, Dongdong 75, 81
Zhang, Hao 4, 201, 204, 205, 209, 233, 301
Zhang, Ji 74
Zhang, Jiacheng 82, 126
Zhang, Jiajun 44, 49, 56, 80, 88
Zhang, Jinfeng 37
Zhang, Jingyi 88, 108
Zhang, Lu 23
Zhang, Min 73, 74, 82, 88

Zhang, Qiang 23, 36
Zhang, Saizheng 68
Zhang, Shaonan 78
Zhang, Shiyue 45, 74, 79
Zhang, Tong 60, 67, 82, 87
Zhang, Wei Emma 69
Zhang, Weinan 68
Zhang, Xiangwen 45, 60, 87
Zhang, Xiangyu 37, 64
Zhang, Xueqiang 74
Zhang, Ying 68
Zhang, Yong 23
Zhang, Yue 170
Zhang, Yujie 49
Zhang, Zheng 72
Zhang, Zhirui 45, 56, 68, 69, 78, 88
Zhang, Zhisong 46
Zhao, Hai 46, 52
Zhao, Jake 45
Zhao, Kai 60, 141
Zhao, Li 56, 68, 69
Zhao, Rui 116
Zhao, Tiejun 81
Zhao, Wei 181
Zhao, Yang 44
Zhao, Yinggong 8, 94, 124, 125, 128, 131
Zheng, Thomas Fang 45, 79
Zheng, Xiaoqiang 29, 124, 128, 148, 198
Zhou, Bowen 36, 49
Zhou, Dengyong 73
Zhou, Guodong 82
Zhou, Long 44, 88
Zhou, Ming 36, 45, 56, 68, 69, 72, 73, 75, 77, 78, 81, 88, 174, 181
Zhou, Tianyi 23, 29, 36, 71
Zhu, Jingbo 49, 60, 81
Zhu, Kenny 61
Zhu, Menglong 34
Zhu, Michael 79
Zhu, Wei-Jing 57, 67, 70
Zhu, Wenhuan 43, 45
Zhukov, Vlad 67
Zipf, George Kingsley 48
Zipser, David 74
Zong, Chengqing 44, 49, 56, 80, 88
Zoph, Barret 44, 67, 77, 79